NILS HARTMANN
https://nilshartmann.net

Frontend-Entwicklung

HTMX oder Single-Page-Anwendung?

NILS HARTMANN

nils@nilshartmann.net

Freiberuflicher Entwickler, Architekt, Trainer

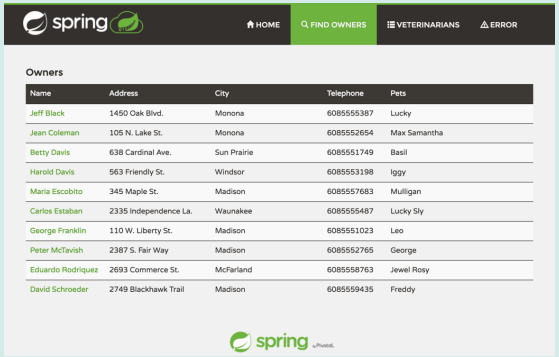
Java, Spring, GraphQL, React, TypeScript

https://graphql.schule/video-kurs

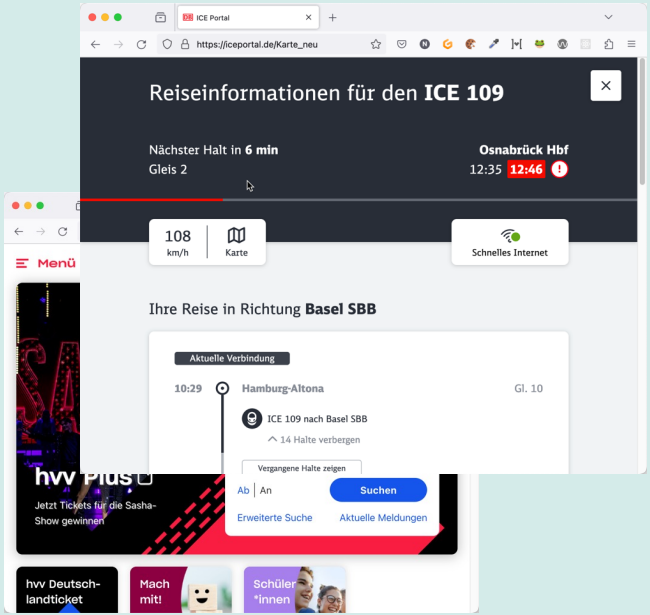
https://reactbuch.de

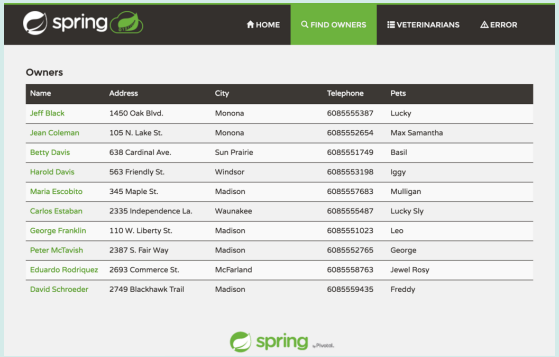
Webanwendungen...

WEBANWENDUNGEN

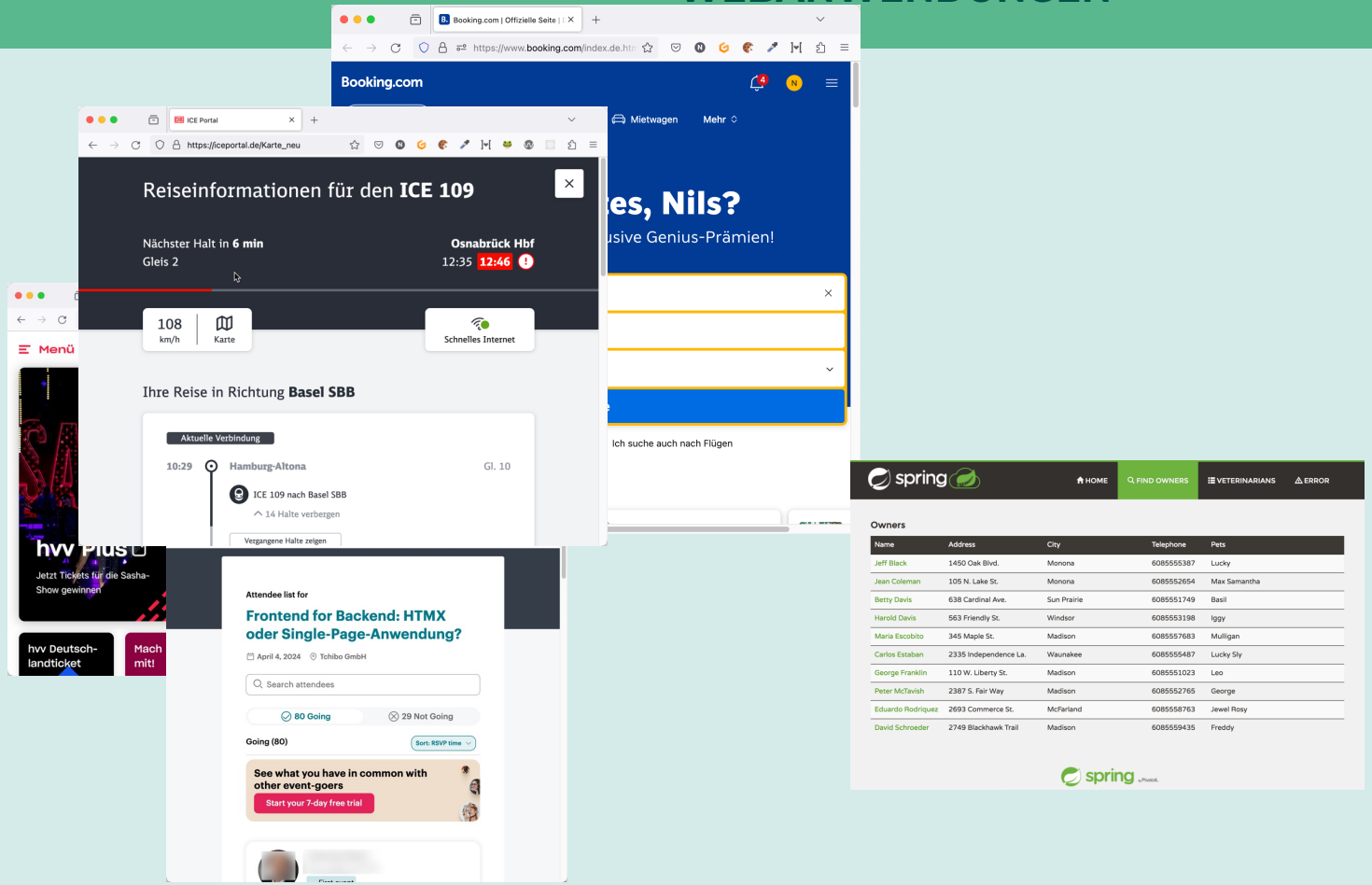


WEBANWENDUNGEN

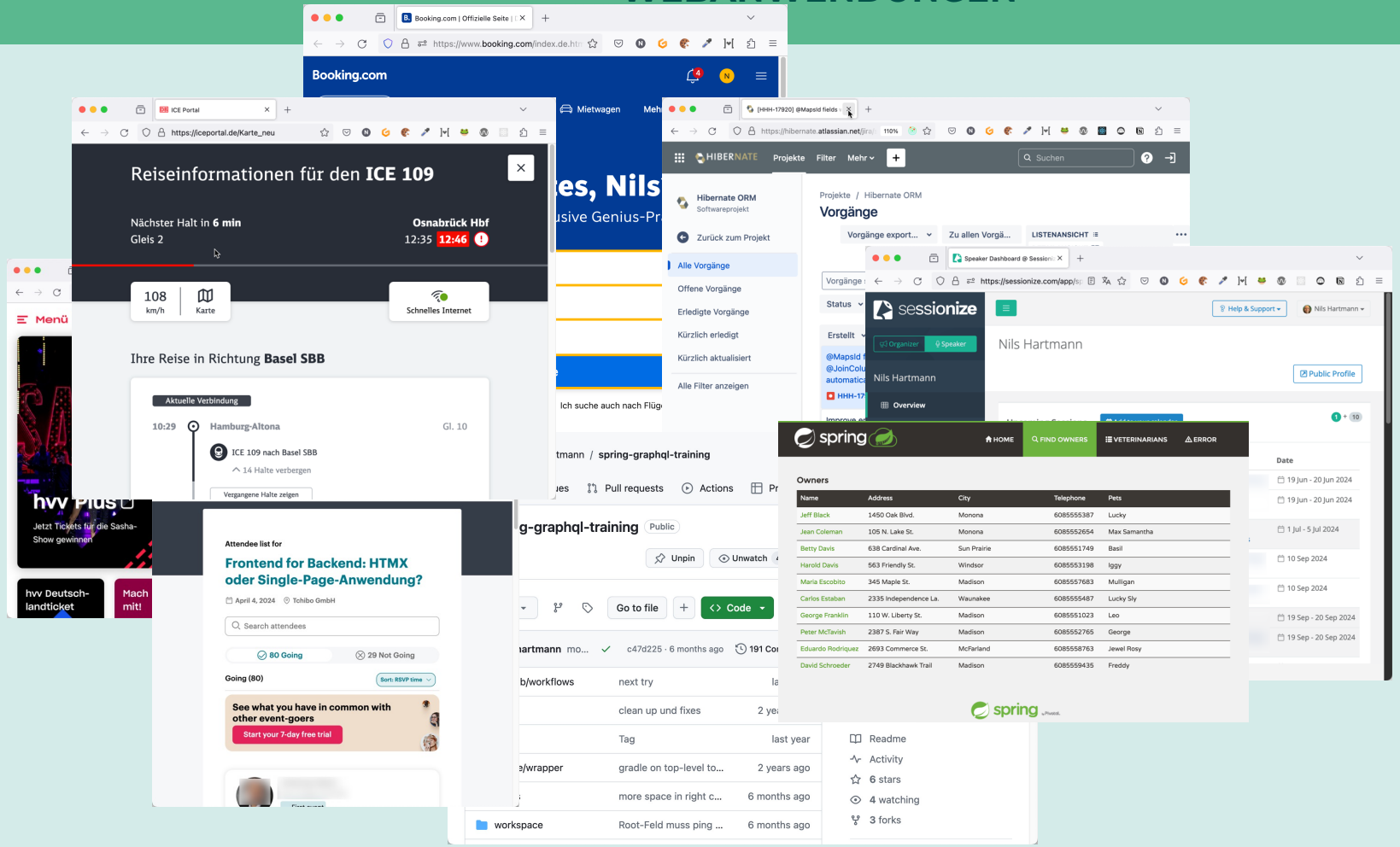




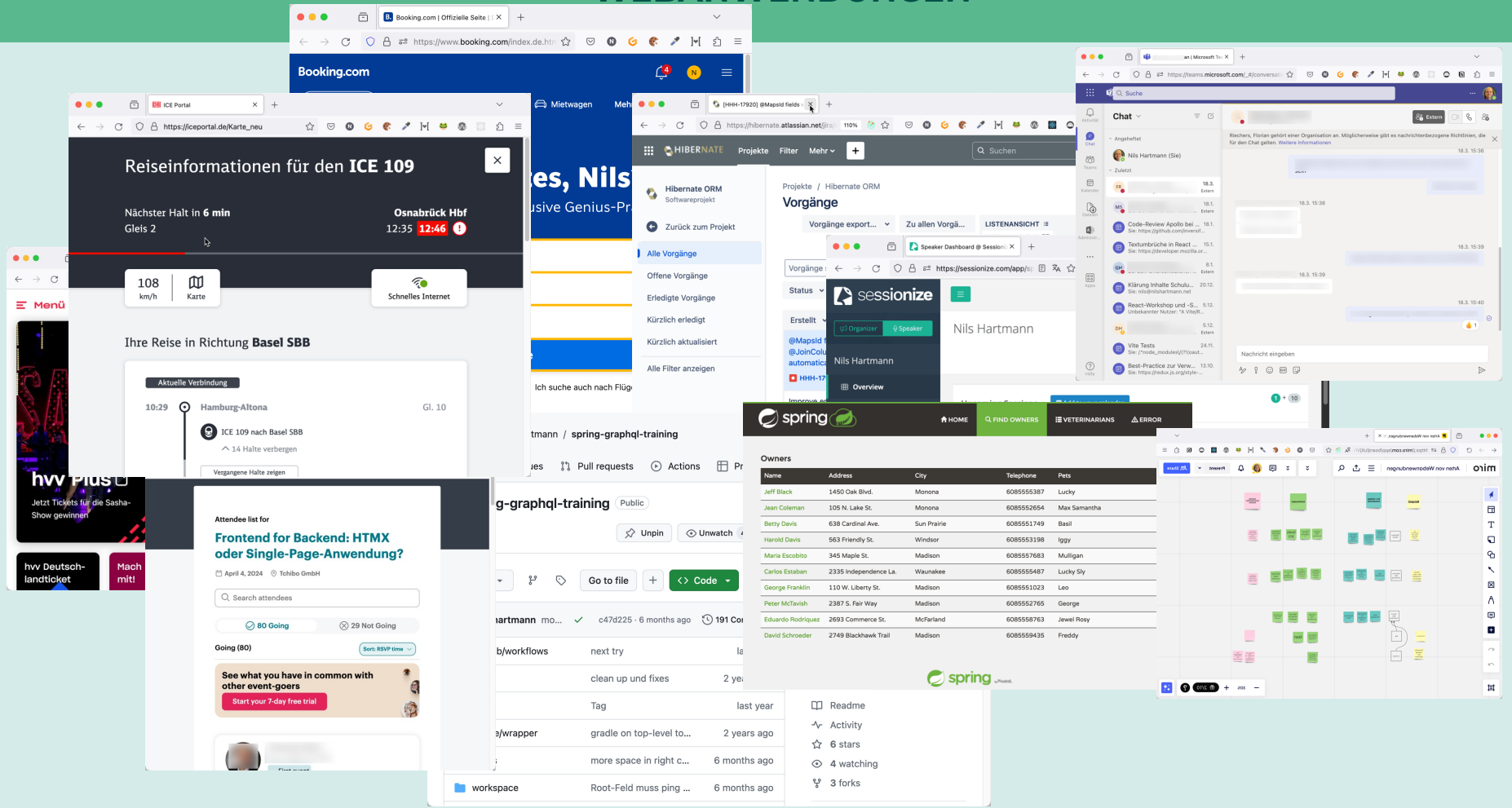
WEBANWENDUNGEN



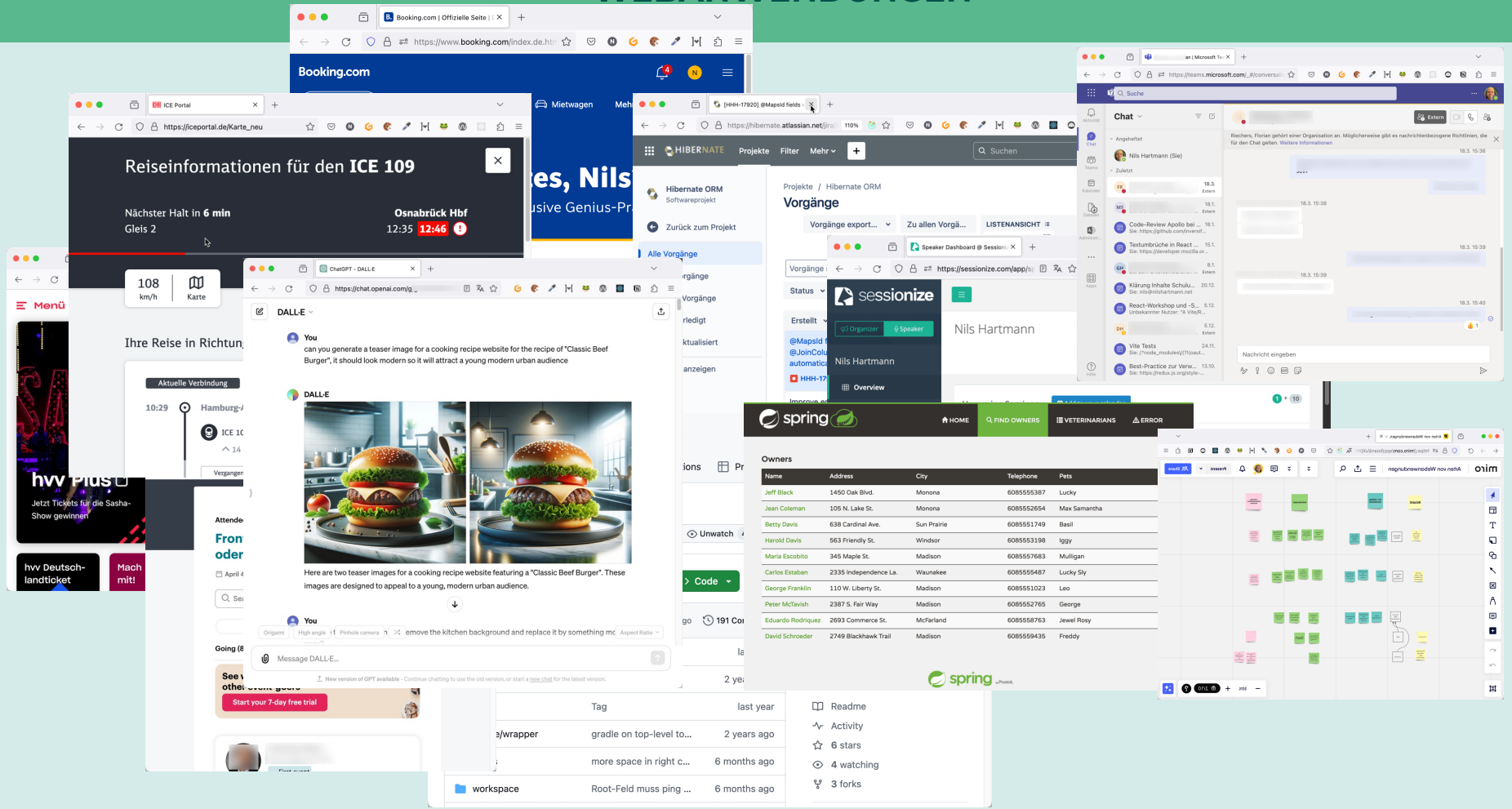
WEBANWENDUNGEN



WEBANWENDUNGEN

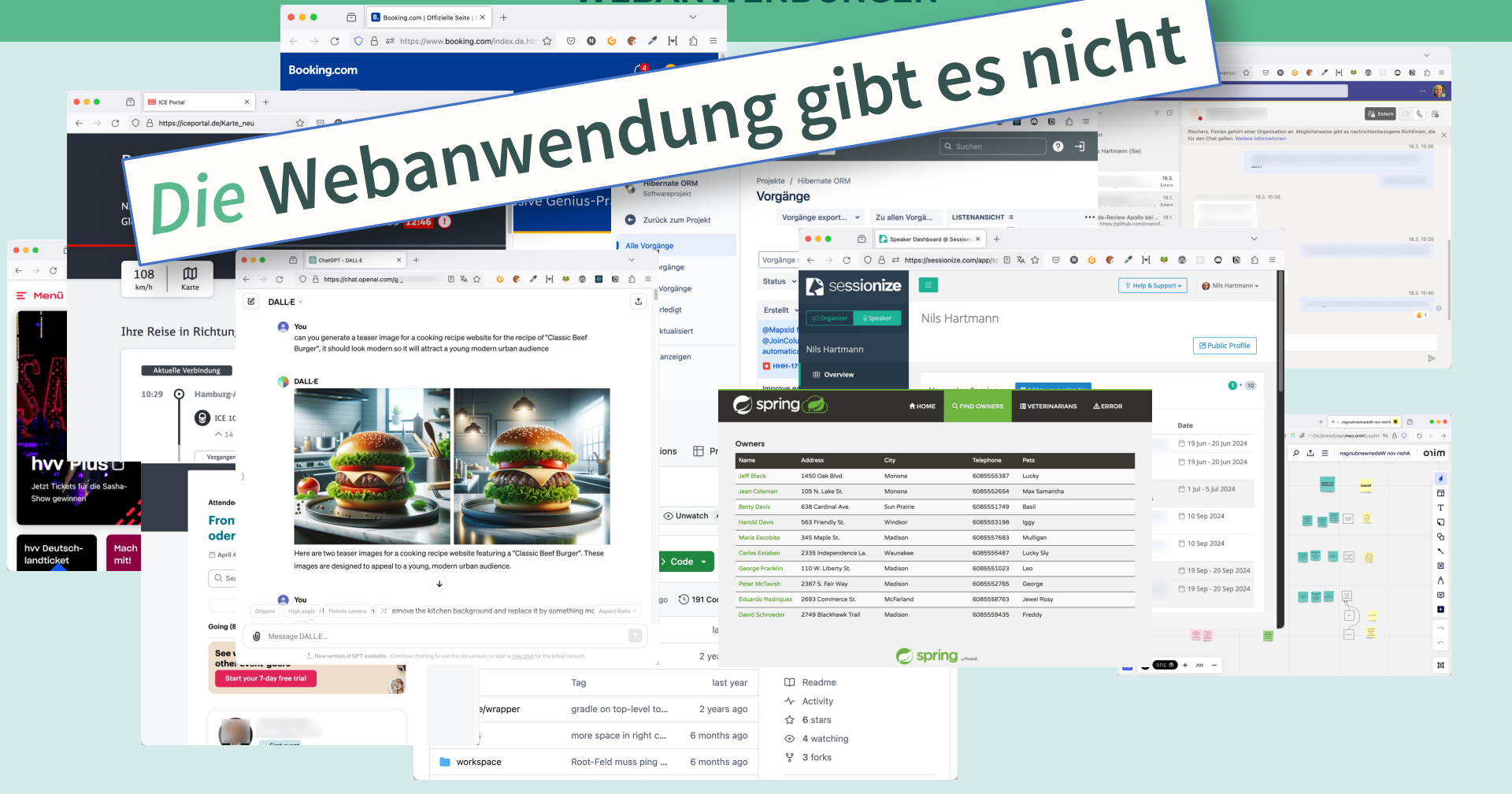


WEBANWENDUNGEN

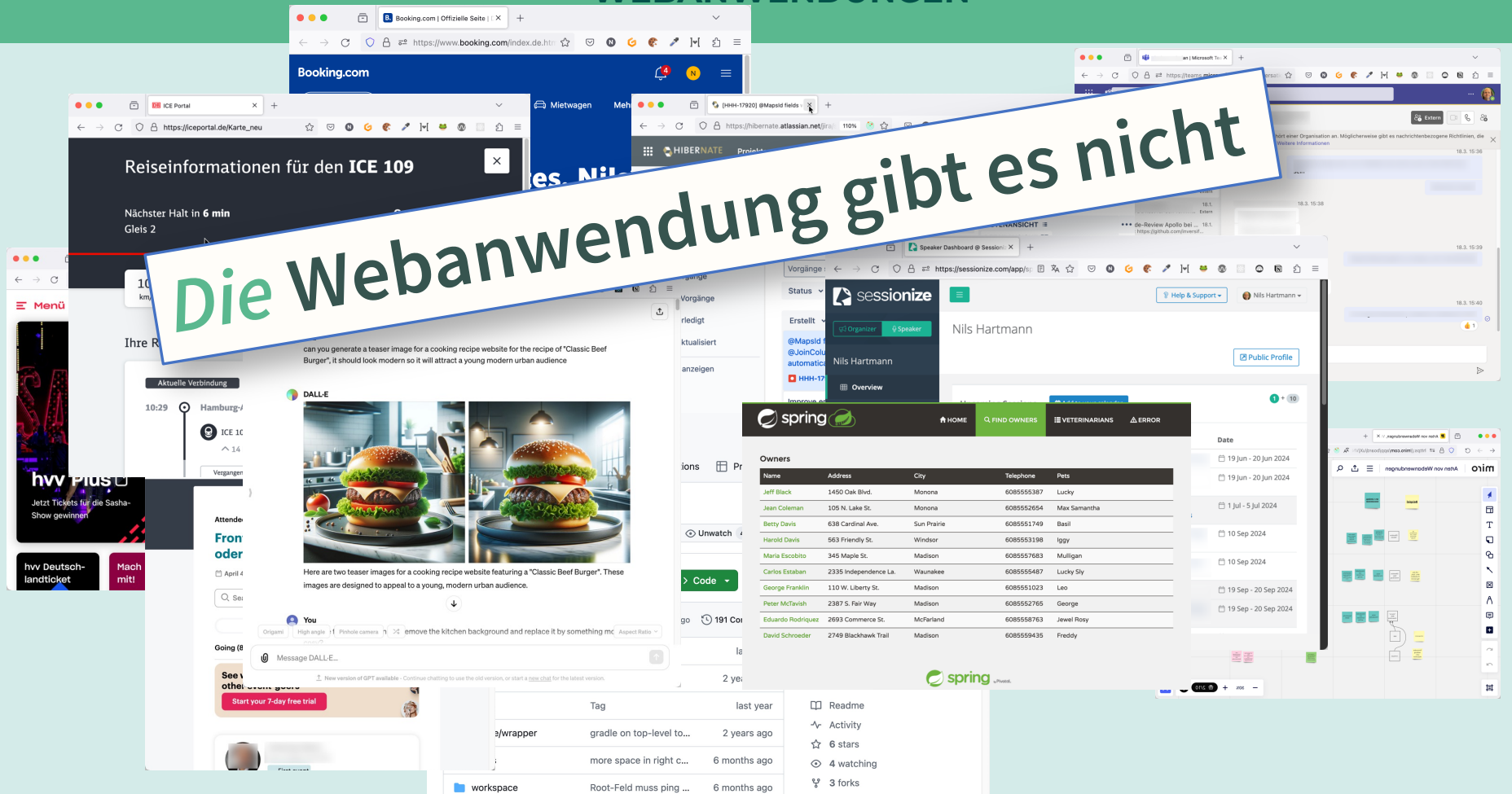


WEBANWENDUNGEN

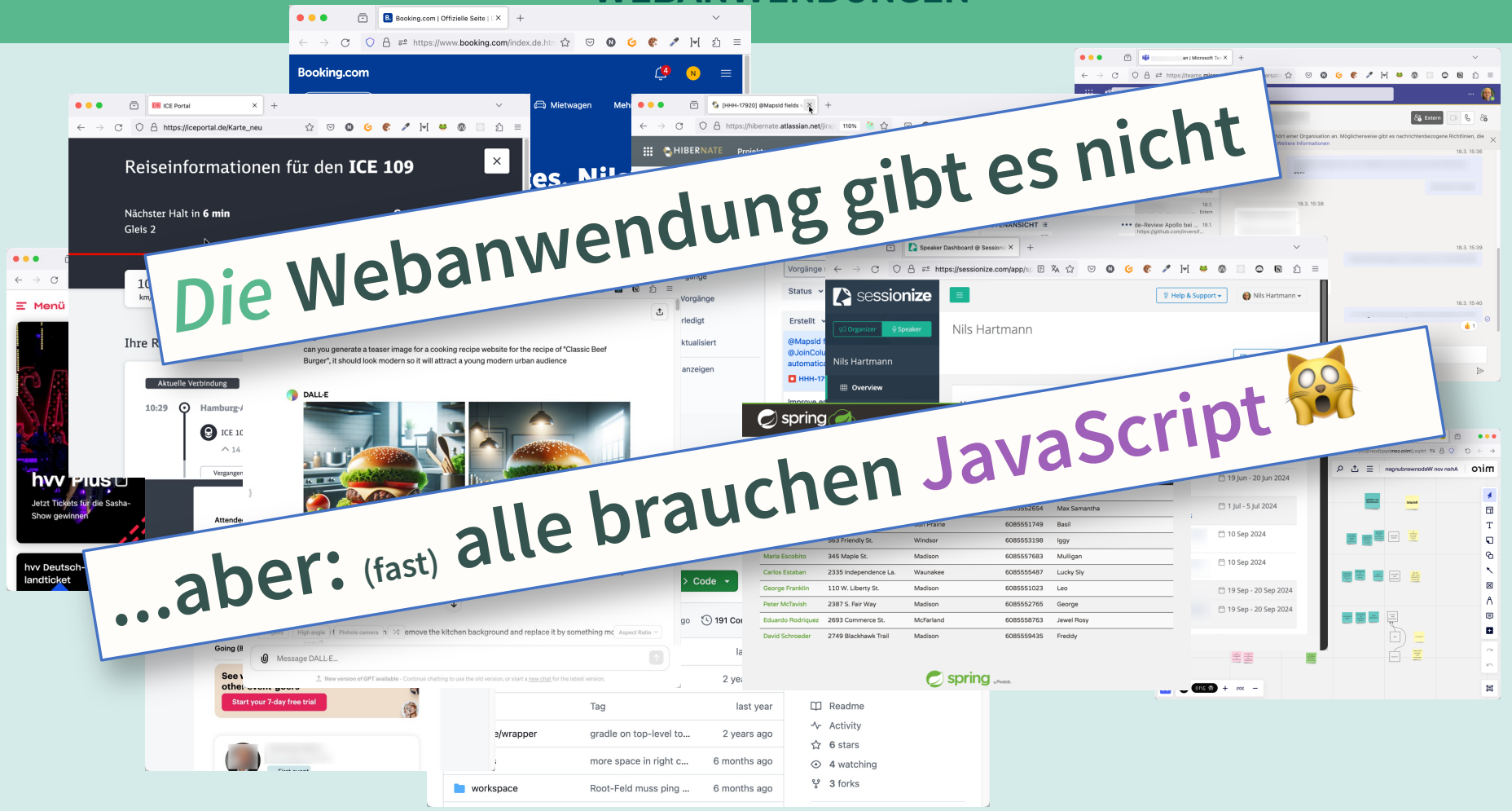
Die Webanwendung gibt es nicht



WEBANWENDUNGEN

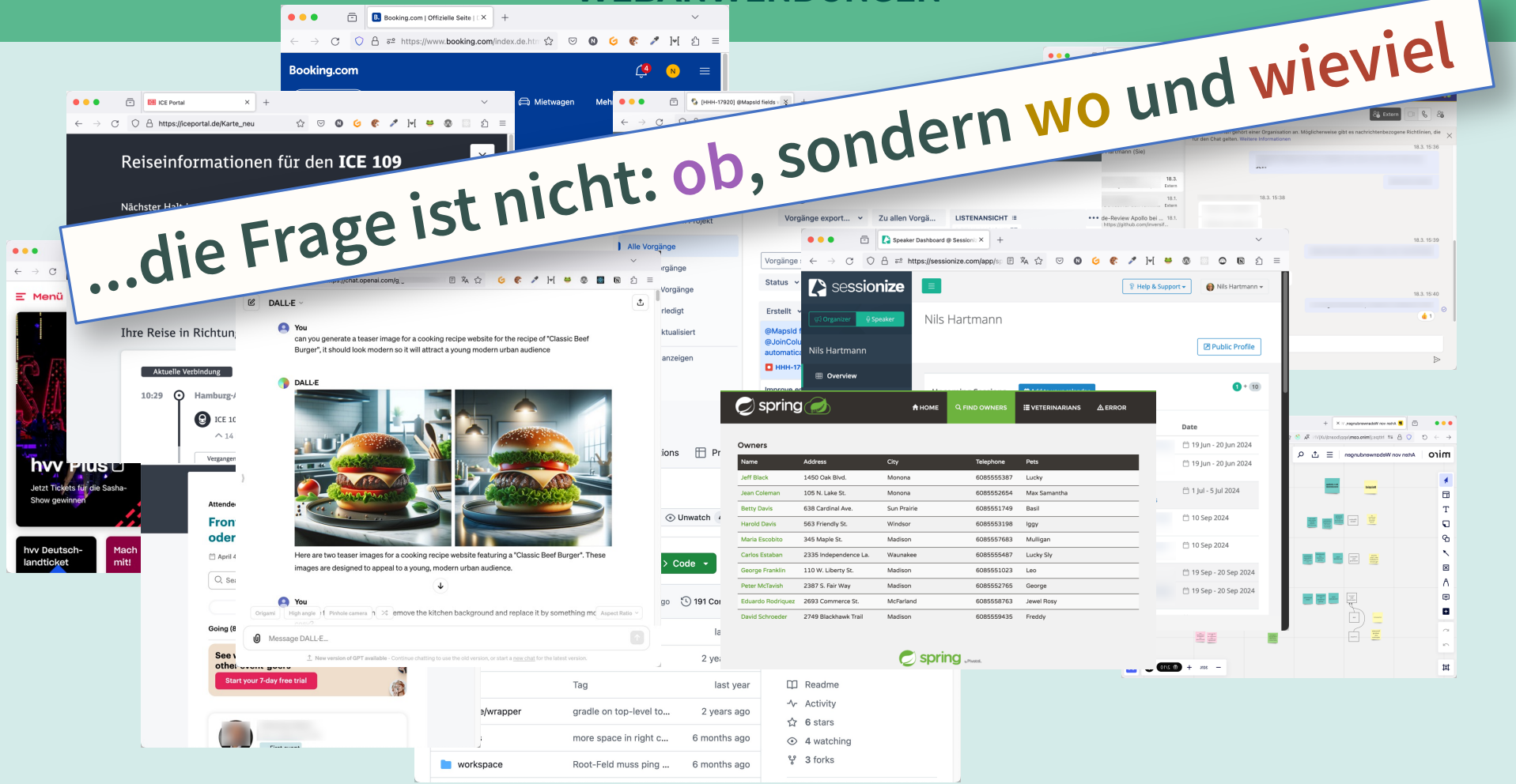


WEBANWENDUNGEN



WEBANWENDUNGEN

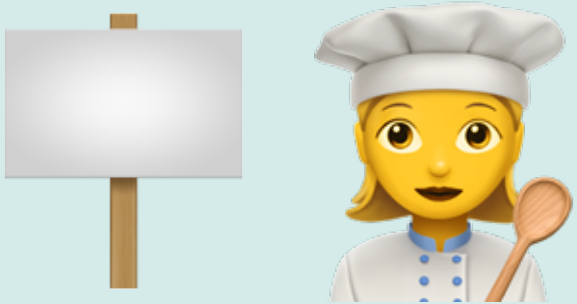
…die Frage ist nicht: ob, sondern wo und wieviel



WEBANWENDUNGEN


...die Frage ist nicht: ob, sondern wo und wieviel
wie gehen wir mit JavaScript um?
darum geht es heute
Booking.com
Reiseinformationen für den ICE 109
DALL·E
You
can you generate a teaser image for a cooking recipe website for the recipe of "Classic Beef Burger", it should look modern so it will attract a young modern urban audience
DALL·E
Here are two teaser images for a cooking recipe website featuring a "Classic Beef Burger". These images are designed to appeal to a young, modern urban audience.
sessionize
Menü
hvv Plus
Jetzt Tickets für die Sasha-Show gewinnen
Aktuelle Verbindung
Start your 7-day free trial
Code
Unwatch
6 stars
4 watching
3 forks
Root-Feld muss ping ...
6 months ago

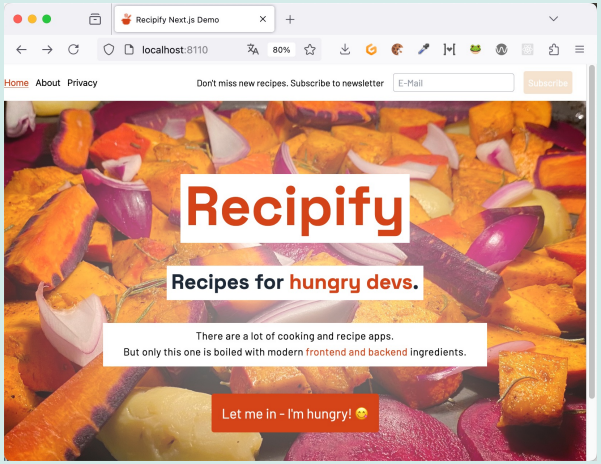
Demo



BEISPIELE

Eine Webanwendung...

1. 🕵️ Landing-Page
2. 🕵️ Recipes
3. 🕵️ Link mit Sortierung
4. 🕵️ Like-Button
5. 🕵️ Recipe-Seite
6. 🕵️ Recipe-Seite Kommentar schreiben



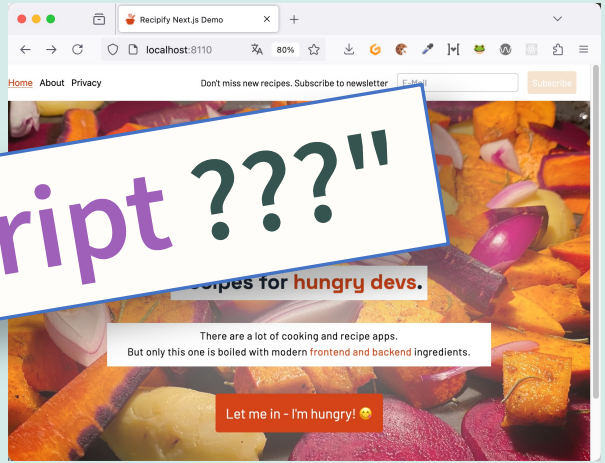
http://localhost:8110

BEISPIELE

Eine Webanwendung...

1. 🕵️ Landing-Page
2. 🕵️ Recipes
3. 🕵️ Rec
4. 🕵️ y
5. 🕵️ Recipe-Seite
6. 🕵️ Recipe-Seite Kommentar schreiben

"alle brauchen JavaScript ???"

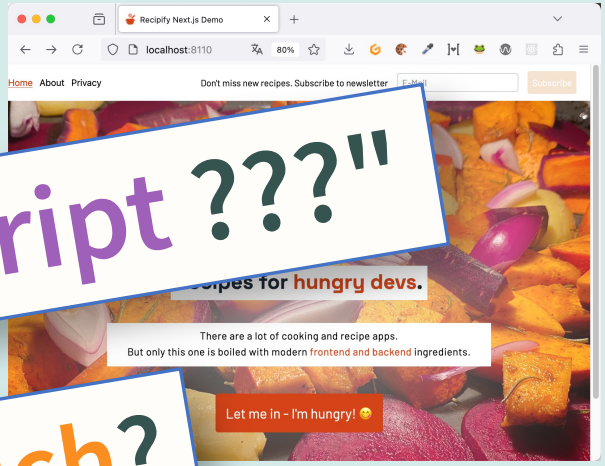


http://localhost:8110

BEISPIELE

Eine Webanwendung...

1. 🕵️ Landing-Page
2. 🕵️ Recipes
3. 🕵️ Rec…
4. …
5. 🕵️ Recipe-Seite
6. 🕵️ Recipe-Se… …schreiben



…//localhost:8110

"alle brauchen JavaScript ???"

diese auch? wirklich?

was meint ihr? 🙋

BEISPIELE

Eine Webanwendung...

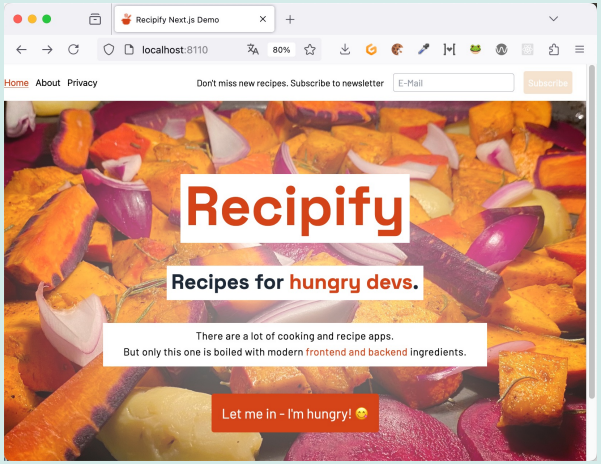
1. 🕵️ Landing-Page
2. 🕵️
3. 🕵️
4. 🕵️ L
5. 🕵️ Re
6. 🕵️ Re

Eine Webanwendung...

1. 🕵️ Landing-Page
2. 🕵️
3. 🕵️
4.
5. 🕵️ Re
6. 🕵️ Re

BEISPIELE

Offenbar geht's doch auch ohne

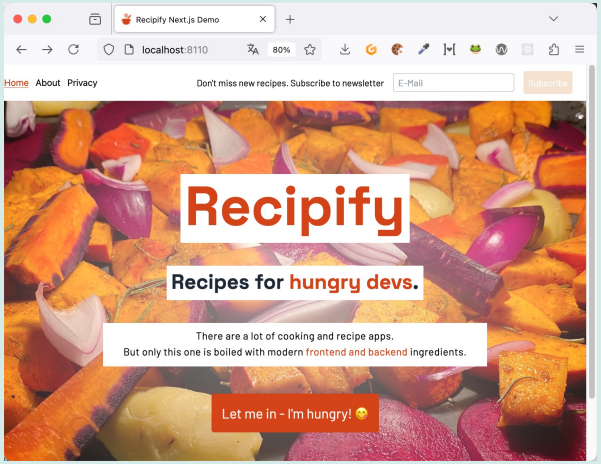


http://localhost:8110

BEISPIELE

Offenbar geht's doch auch ohne – ABER:

1. 🕵️ Feedback Like-Button 😢
2. 🕵️ Feedback Seitenwechsel zu Recipe 😢
3. 🕵️ Timer 😢
4. 🕵️ Newsletter-Feld Inhalt Seitenwechsel 😢
5. 🕵️ Neuer Kommentar: Zeichenzähler 😢
6. 🕵️ Portionszähler 😢
7. 🕵️ Suche bei Tastendruck 😢



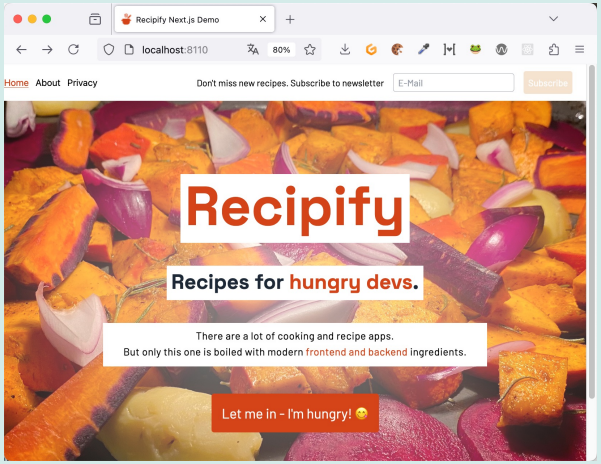
http://localhost:8110

BEISPIELE

Wie sieht es denn mit JavaScript aus?

```
demo_config: enableSuspenseForFeedback = true
```

1. 🕵️ Feedback Like-Button 😊
2. 🕵️ Feedback Seitenwechsel zu Recipe 😊
3. 🕵️ Timer 😊
4. 🕵️ Newsletter-Feld Inhalt Seitenwechsel 😊
5. 🕵️ Neuer Kommentar: Zeichenzähler 😊
6. 🕵️ Portionszähler 😊
7. 🕵️ Suche bei Tastendruck 😊



http://localhost:8110

Beste UI/UX: oft nur mit JavaScript umsetzbar

- Es gibt Fortschritte in CSS und HTML, die JS an mehr Stellen verzichtbar machen, aber nicht an allen

Beste UI/UX: oft nur mit JavaScript umsetzbar

- Es gibt Fortschritte in CSS und HTML, die JS an mehr Stellen verzichtbar machen, aber nicht an allen

Man sollte den "Kleinkram" nicht unterschätzen 🙏

Architekturen

für

Webanwendungen

Ansatz 1: "Klassisch": Serverseitiges Rendern + JS-Schnipsel

- Server erzeugt fertige HTML-Seite
- Für Interaktionen werden Links und Formulare genommen
- Jede Interaktion erfolgt über Server-Roundtrip

Ansatz 1: "Klassisch": Serverseitiges Rendern + JS-Schnipsel

- Server erzeugt fertige HTML-Seite
- Für Interaktionen werden Links und Formulare genommen
- Jede Interaktion erfolgt über Server-Roundtrip
- Für Interaktionen wird JavaScript geschrieben

Ansatz 1: "Klassisch": Serverseitiges Rendern + JS-Schnipsel

- Server erzeugt fertige HTML-Seite
- Für Interaktionen werden Links und Formulare genommen
- Jede Interaktion erfolgt über Server-Roundtrip
- Für Interaktionen wird JavaScript geschrieben

- **Typische Vertreter**: **Spring MVC**, JEE

Ansatz 2: Single-Page-Anwendung (SPA)

- Darstellung erfolgt komplett über JavaScript im Browser
- HTML spielt untergeordnete Rolle
- Backend liefert **Daten** (z.B. REST/GraphQL), kein HTML

Ansatz 2: Single-Page-Anwendung (SPA)

- Darstellung erfolgt komplett über JavaScript im Browser
- HTML spielt untergeordnete Rolle
- Backend liefert **Daten** (z.B. REST/GraphQL), kein HTML

- **Typische Vertreter:** **React**, Angular, Vue

Ansatz 3: Fullstack-Anwendung (mit JavaScript)

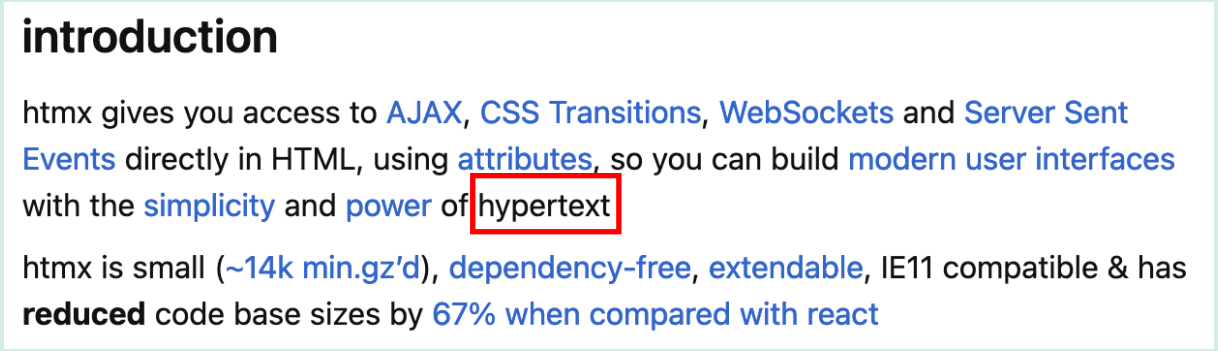
- Anwendung ist vollständig in JavaScript geschrieben
- Server läuft mit JavaScript
- Server liefert HTML-Code aus
- Server liefert für interaktive Teile JavaScript-Code aus
- Zum Beispiel als "Backend for Frontend"

Ansatz 3: Fullstack-Anwendung (mit JavaScript)

- Anwendung ist vollständig in JavaScript geschrieben
- Server läuft mit JavaScript
- Server liefert HTML-Code aus
- Server liefert für interaktive Teile JavaScript-Code aus
- Zum Beispiel als "Backend for Frontend"
- **Typische Vertreter: Next.js**, SvelteKit, Nuxt, Astro, Qvik

...und HTMX?

HTMX ist eine JavaScript-Bibliothek



introduction

htmx gives you access to AJAX, CSS Transitions, WebSockets and Server Sent Events directly in HTML, using attributes, so you can build modern user interfaces with the simplicity and power of hypertext

htmx is small (~14k min.gz'd), dependency-free, extendable, IE11 compatible & has **reduced** code base sizes by 67% when compared with react

https://htmx.org/

Versprechen: Aktualisierung der Darstellung ohne selbst JavaScript **schreiben** zu müssen

(stattdessen "hypertext" 🤔)

Passt zur Variante 1 (serverseitige Anwendungen)

HTMX - Grundlagen

- HTML-Elemente werden mit HTMX-Attributen ergänzt
- Damit wird beschrieben, welche Server Requests bei einem "Trigger" ausgeführt werden sollen
- HTMX kümmert sich um die Ausführung des Requests und die Verarbeitung der Antwort
- Andere Aktualisierungen der UI gehen mit HTMX nicht

```
<html lang="en">
  <body>
    <div hx-get="/hello-world"
         hx-trigger="click"
         hx-target="#result">
      Get Greeting
    </div>

    <div id="result"></div>

    <script
      type="text/javascript"
      th:src="@{/htmx/htmx-1.9.10.min.js}"
    ></script>
  </body>
</html>
```

HTMX – Der Server

- Der Server nimmt (Ajax-)Requests entgegen
- HTMX-Requests sind per HTTP Header zu identifizieren
- Der Server liefert dann HTML-Schnipsel
- Wie die Schnipsel erzeugt werden, entscheidet der Server frei

```
<html lang="en">
  <body>
    <div hx-get="/hello-world"
         hx-trigger="click"
         hx-target="#result">
      Get Greeting
    </div>

    <div id="result"></div>

    <script
      type="text/javascript"
      th:src="@{/htmx/htmx-1.9.10.min.js}"
    ></script>
  </body>
</html>
```

HTMX – Grundlagen

- HTML-Elemente werden mit HTMX-Attributen er
- Da
  Re
  wo
- HT
  Re
- De
  ("Hypermedia")

Demo: Hello World

localhost:8080/hello

- hello-world.jte
- HelloWorldController
- hello-world-response.jte

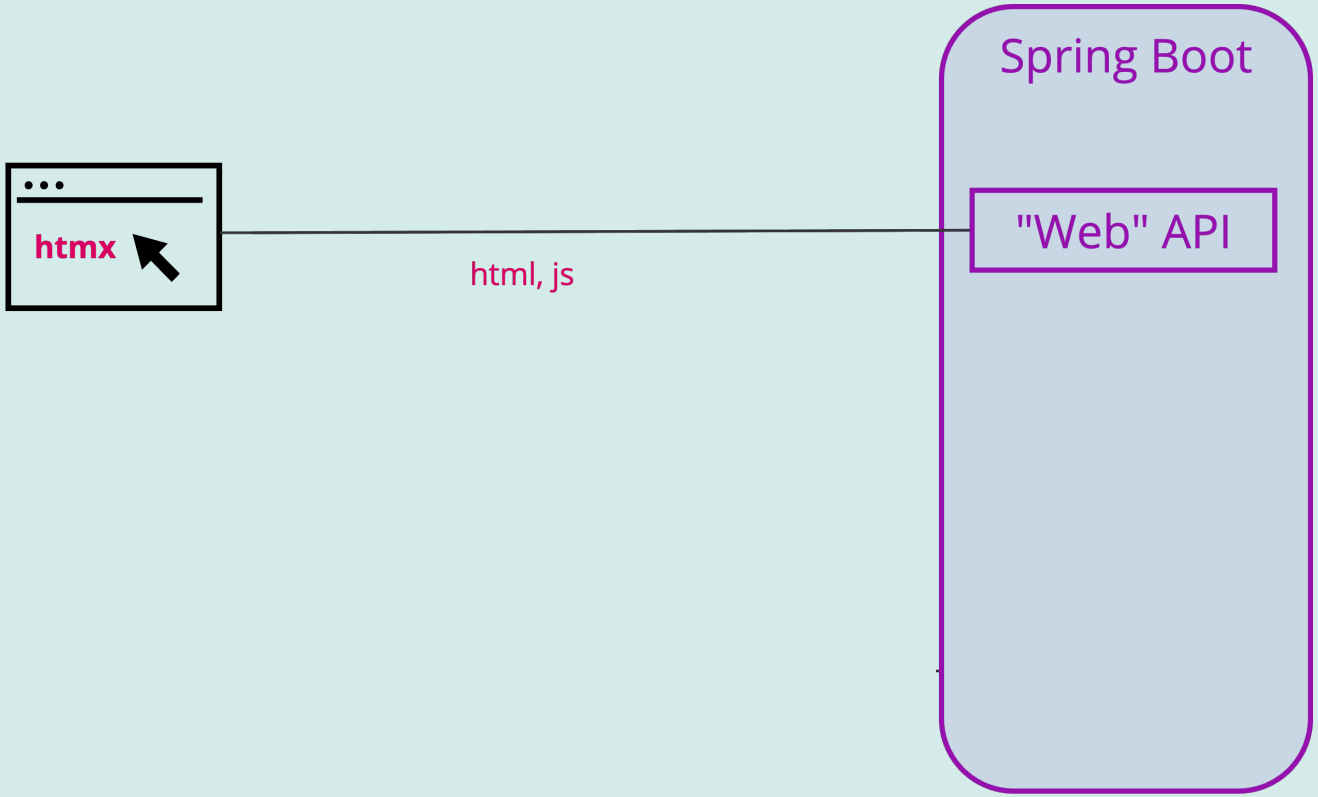
```
<html lang="en">
  <body>
    <div hx-get="/hello-world"
         hx-trigger="click"
         hx-target="#result">
      Get Greeting
    </div>

    <div id="result"></div>

    <script
      type="text/javascript"
      th:src="@{/htmx/htmx-1.9.10.min.js}"
    ></script>
  </body>
</html>
```

Code

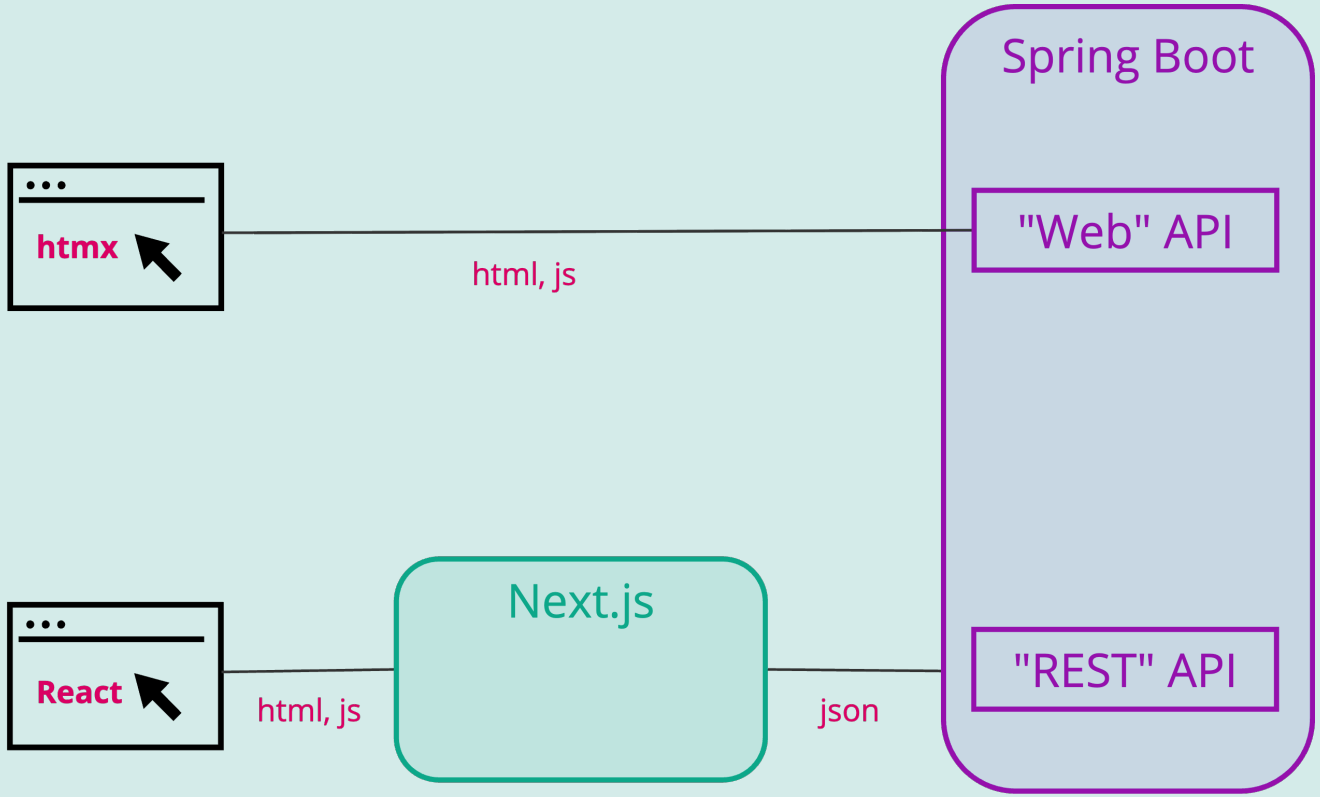
Beispiele

HTTP://LOCALHOST:8080

Architektur und Techstack Recipify



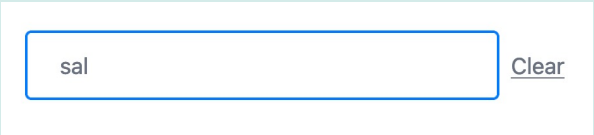
Architektur und Techstack Recipify



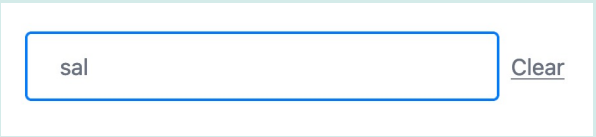
Beispiel: Formulare

- Wie werden die Daten gesendet?
- Was kommt als Antwort?
- Wie disablen wir das Formular?
- Wie setzen wir eine (Fehler-)Meldung zurück
- 🕵️ `NewsletterRegistration.jte / .tsx`

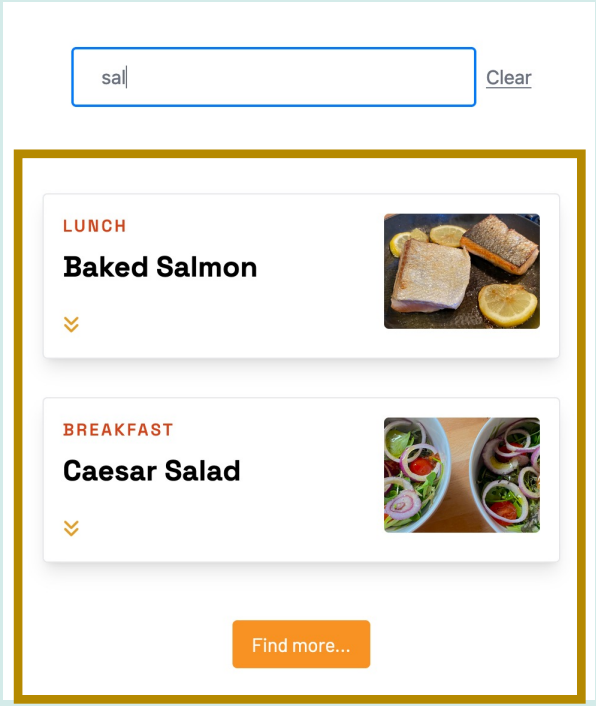
Beispiel: Bestehende Darstellung aktualisieren



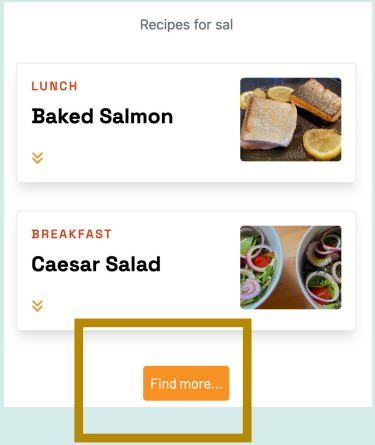
Beispiel: Bestehende Darstellung aktualisieren



GET /search

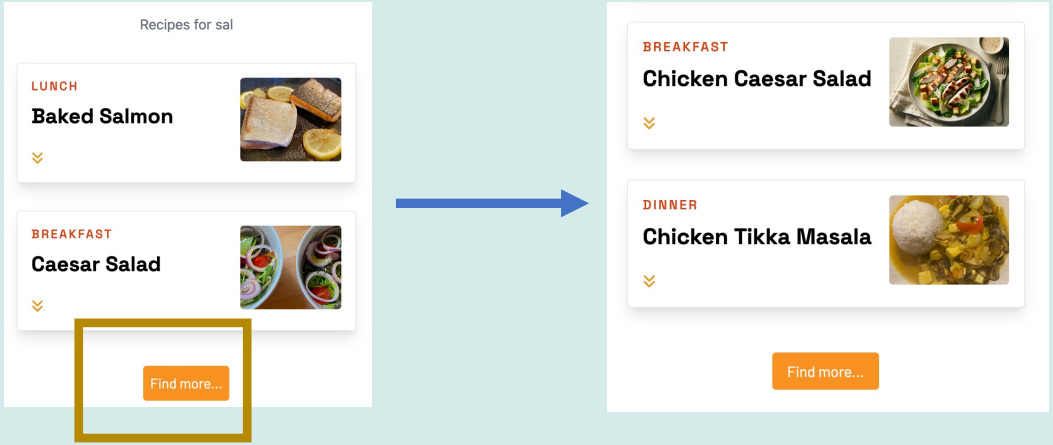


Beispiel: Bestehende Darstellung aktualisieren




```
GET /search?page=2
```

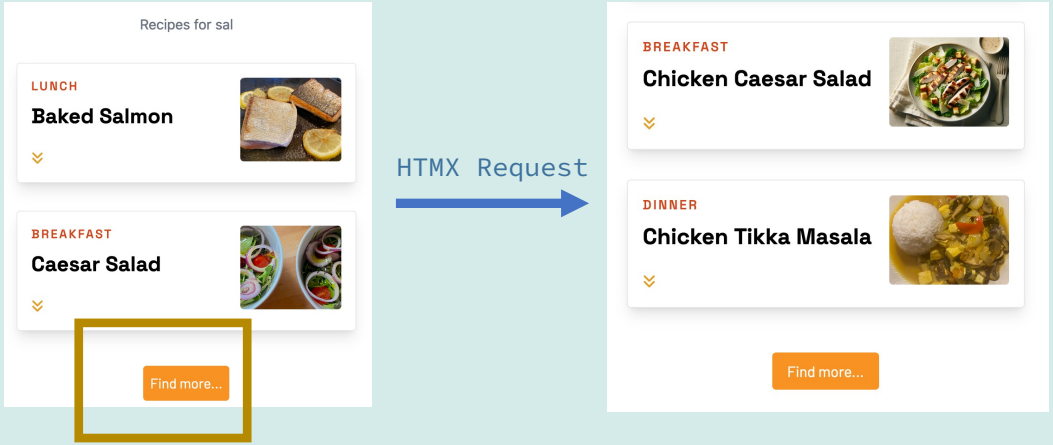
Beispiel: Bestehende Darstellung aktualisieren



`GET /search?page=2`

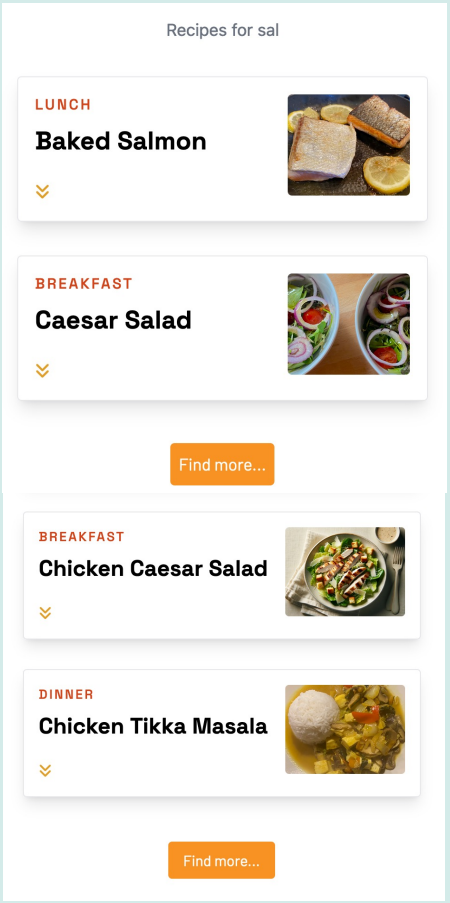
`hx-swap="beforeend"`

Beispiel: Bestehende Darstellung aktualisieren

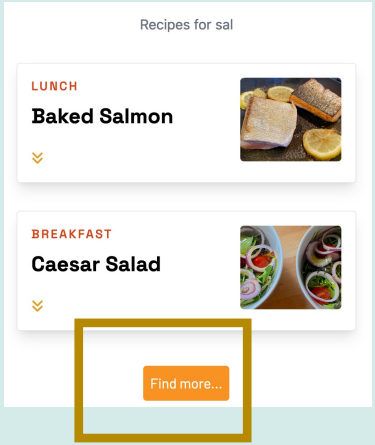


`GET /search?page=2`

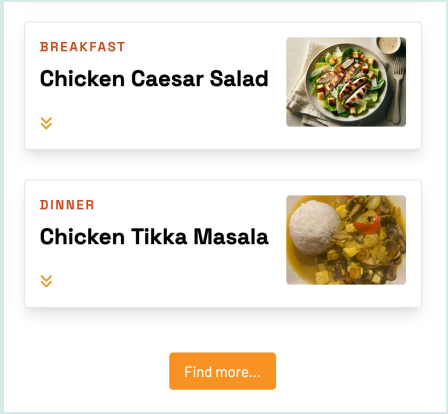
`hx-swap="beforeend"`



Beispiel: Bestehende Darstellung aktualisieren

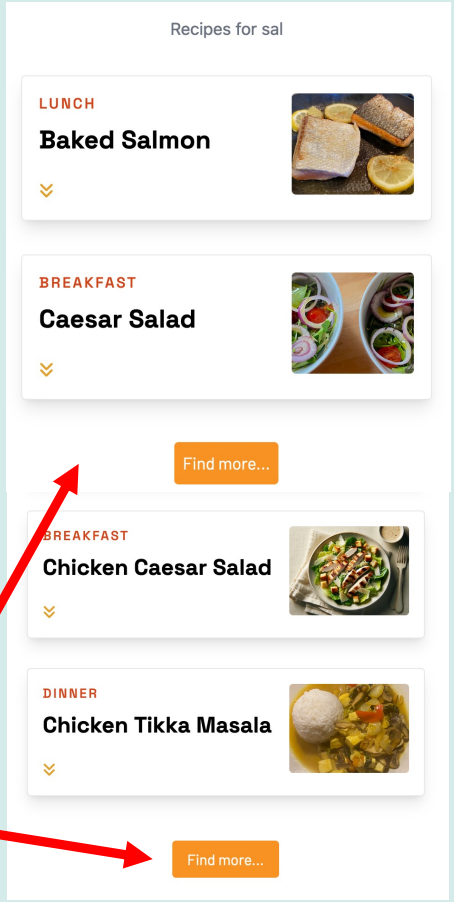


HTMX Request

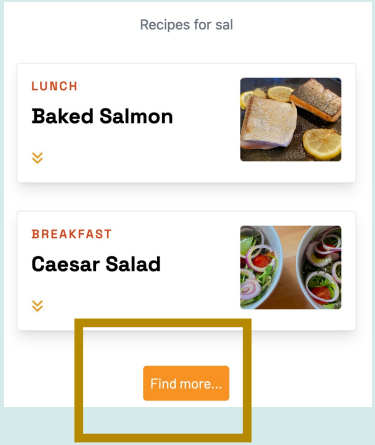


`GET /search?page=2`

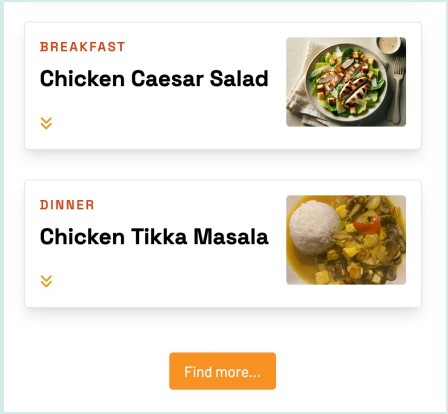
`hx-swap="beforeend"`



Beispiel: Bestehende Darstellung aktualisieren

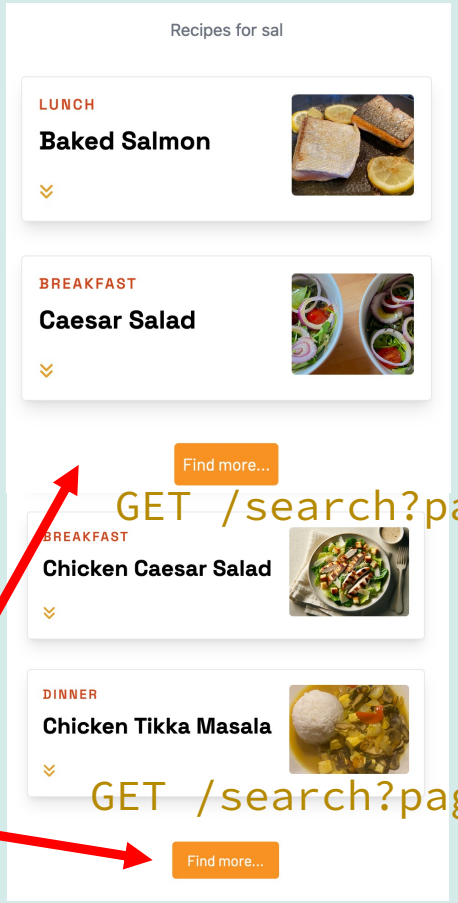


HTMX Request

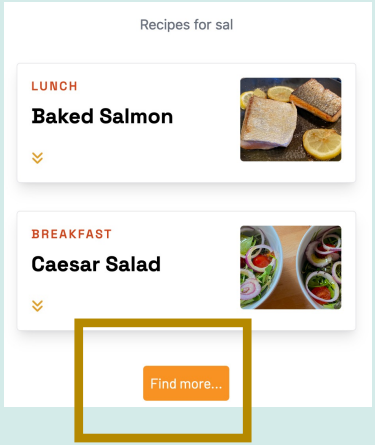


GET /search?page=2

hx-swap="beforeend"

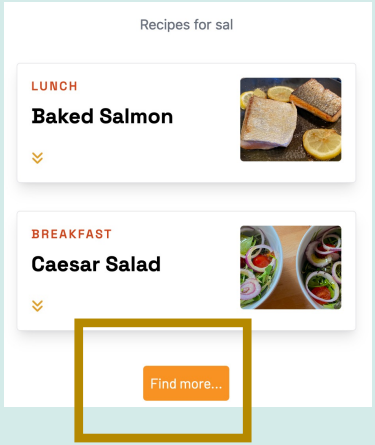


Beispiel: Bestehende Darstellung aktualisieren

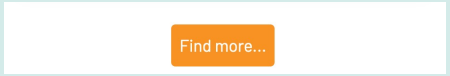



```
GET /search?page=2
```

Beispiel: Bestehende Darstellung aktualisieren

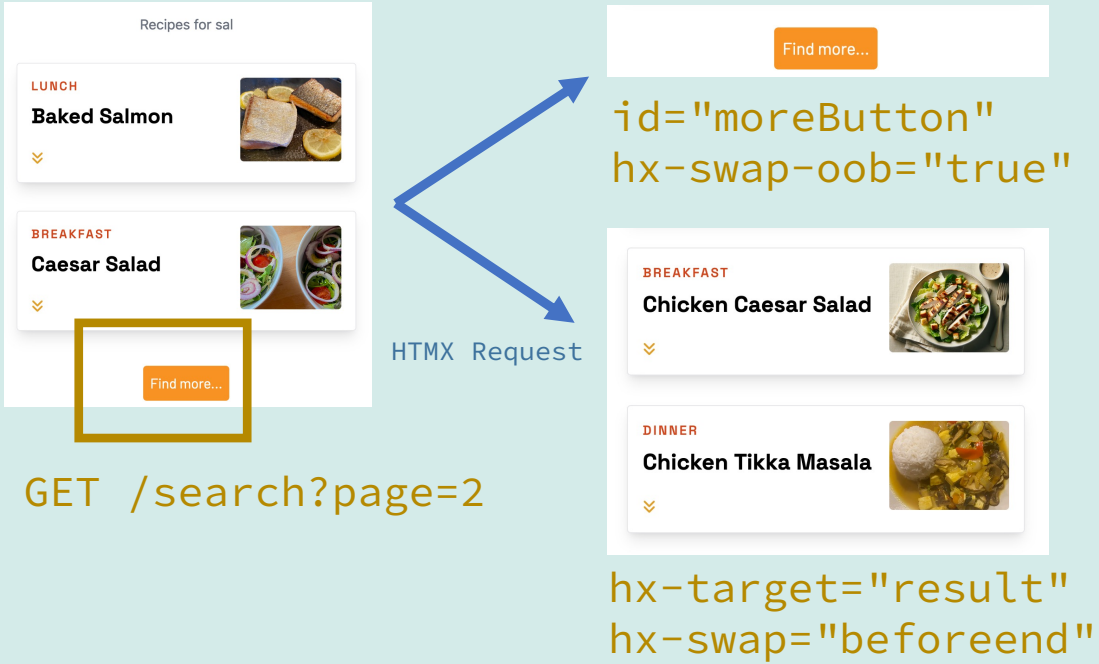



```
GET /search?page=2
```

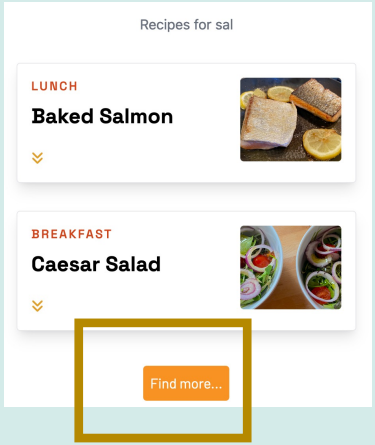



```
id="moreButton"
hx-swap-oob="true"
```

Beispiel: Bestehende Darstellung aktualisieren

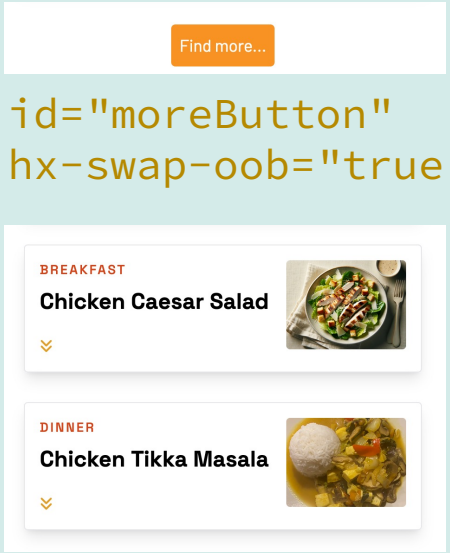


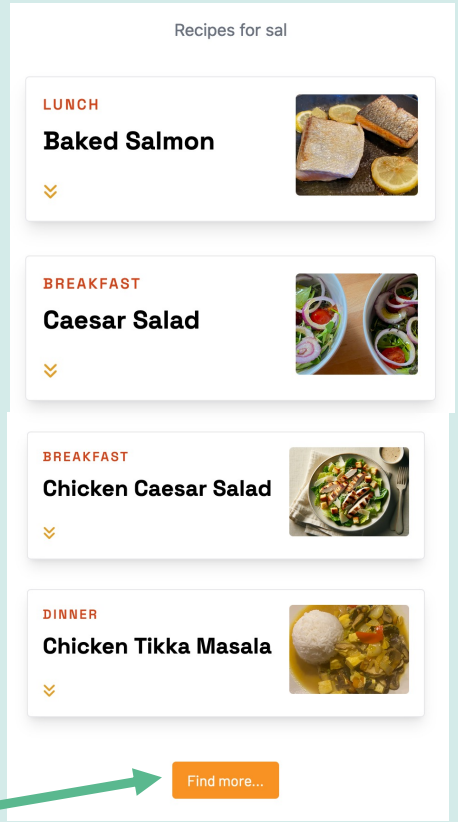
Beispiel: Bestehende Darstellung aktualisieren



GET /search?page=2

HTMX Request

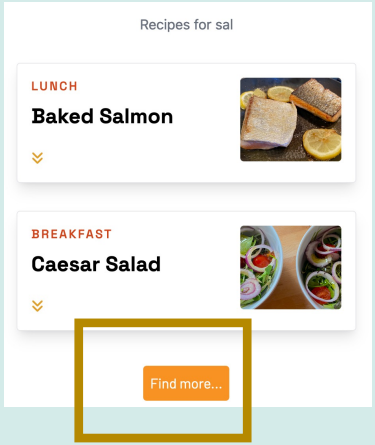




GET /search?page=3

Beispiel: Bestehende Darstellung aktualisieren

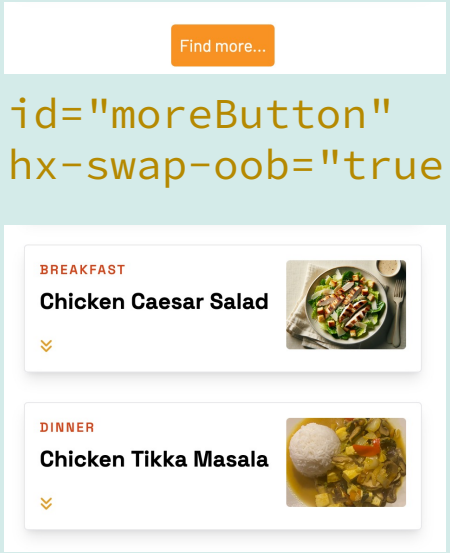
Beispiel: Bestehende Darstellung aktualisieren

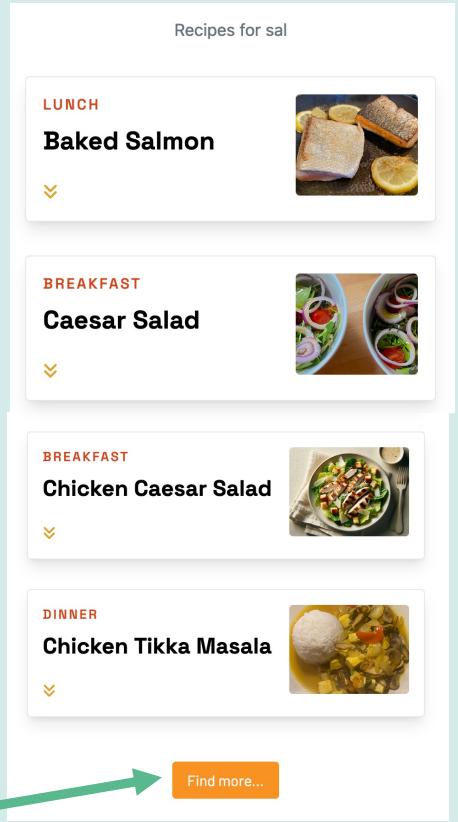


GET /search?page=2

FindMoreButton.jte

RecipeSearch.tsx

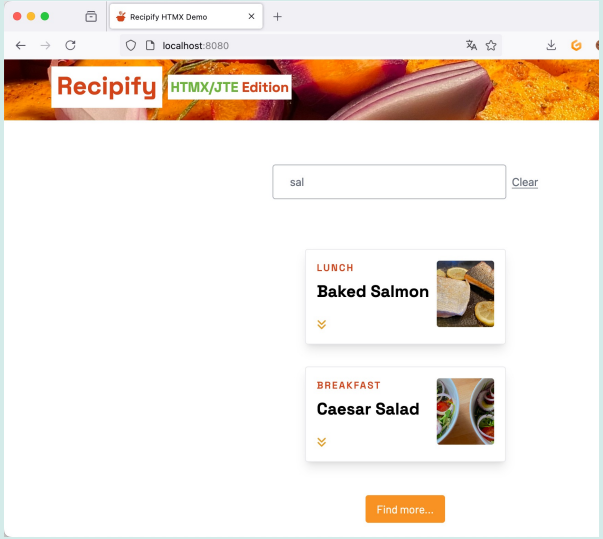




GET /search?page=3

Beispiel: Parallele Requests

- 🕵️ Netzwerk-Tab

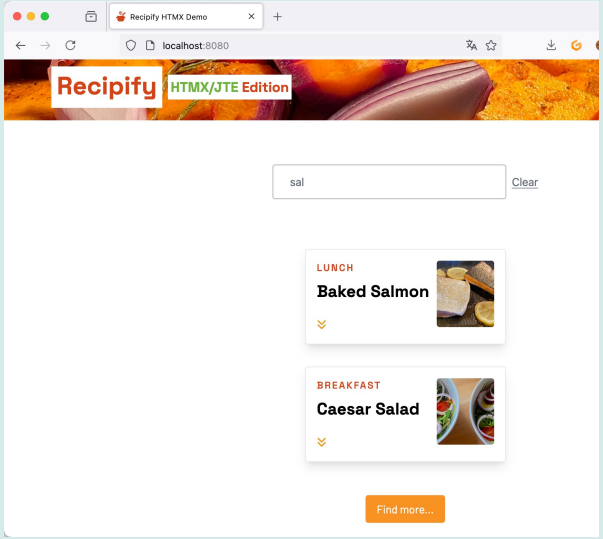



```
SearchPageContent.jte: slowdown=2000, dafür delay raus
```

```
demo-config: slowDown_search=2000, debounce_search=0
```

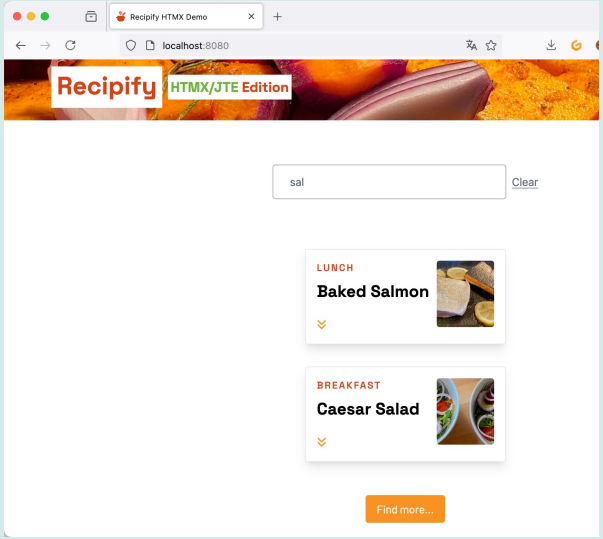
Beispiel: Parallele Requests

- 🕵️ Netzwerk-Tab
- HTMX führt Requests nacheinander aus
- In der SPA können wir das selbst entscheiden



Beispiel: Parallele Requests

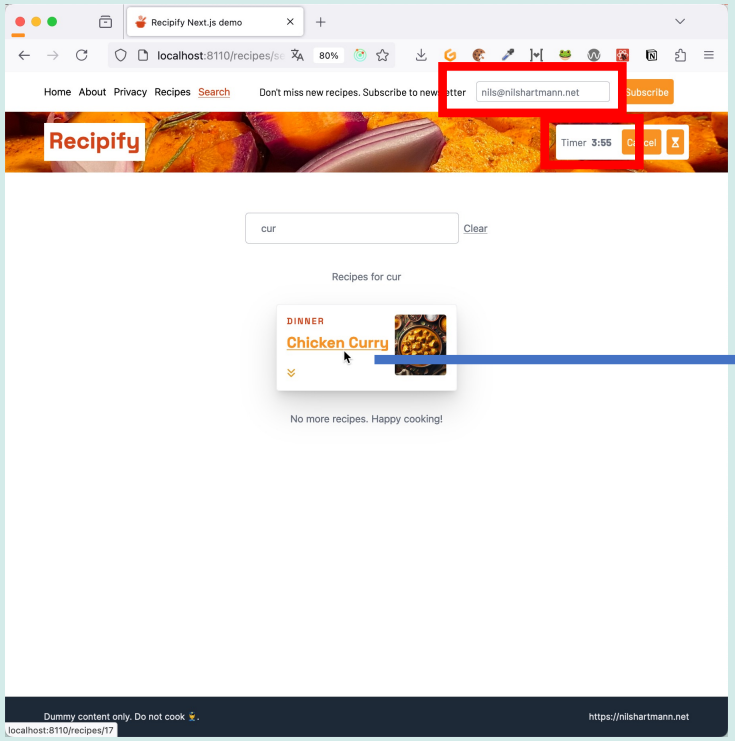
- 🕵️ Netzwerk-Tab
- HTMX führt Requests nacheinander aus
- "Alte" Request werden nicht verworfen
- In der SPA können wir das selbst entscheiden



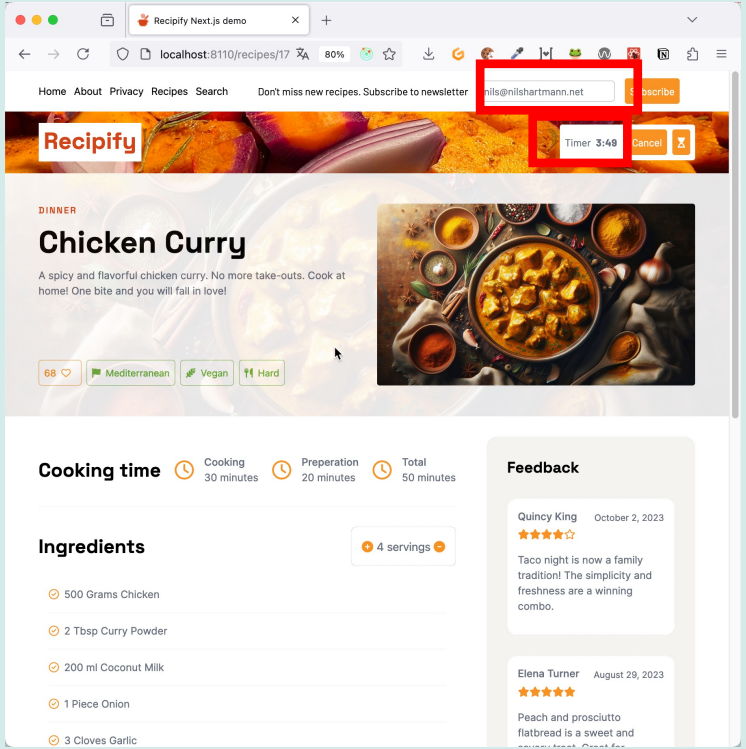
`SearchPageContent.jte: slowdown`

Beispiel: SPA-like Seitenwechsel

- Beim Aufruf der Rezept-Seite soll der Rahmen unverändert bleiben



/recipes/17



Beispiel: SPA-like Seitenwechsel in Next.js

- **Link** verhält sich automatisch richtig:
  - mit und ohne JS
  - Ganze Seite vs. nur Content
  - Aktualisieren der URL

```tsx
<Link
  href={`/recipes/${recipe.id}`}
>
  {recipe.title}
</Link>
```

(RecipeSummaryCard.tsx)

Beispiel: SPA-like Seitenwechsel in Next.js

- **Link** verhält sich automatisch richtig:
  - mit und ohne JS
  - Ganze Seite vs. nur Content
  - Aktualisieren der URL

```
<Link
  href={`/recipes/${recipe.id}`}
>
  {recipe.title}
</Link>
```

(RecipeSummaryCard.tsx)

- "Controller": Identischer Code
  - Fullpage Request
  - Incremental Update

`[recipeId]/page.tsx`

`recipes/layout.tsx`

Beispiel: SPA-like Seitenwechsel in Next.js

- **Link** verhält sich automatisch richtig:
  - mit und ohne JS
  - Ganze Seite vs. nur Content
  - Aktualisieren der URL

```
<Link
  href={`/recipes/${recipe.id}`}
>
  {recipe.title}
</Link>
```

(RecipeSummaryCard.tsx)

- "Controller": Identischer Code
  - Fullpage Request
  - Incremental Update

`[recipeId]/page.tsx`

`recipes/layout.tsx`

- Das funktioniert clientseitig mit allen SPA-Routern

Beispiel: SPA-like Seitenwechsel in HTMX

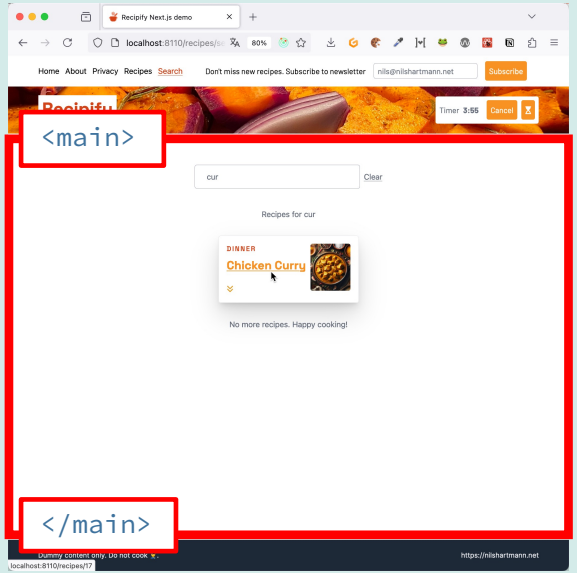
- **Link** muss entsprechend ausgezeichnet werden
  - Für Klick ohne JavaScript
  - Für Klick mit JavaScript
  - Aktualisieren der URL im Browser

`RecipeSummaryCard.jte`

Beispiel: SPA-like Seitenwechsel in HTMX

Variante 1

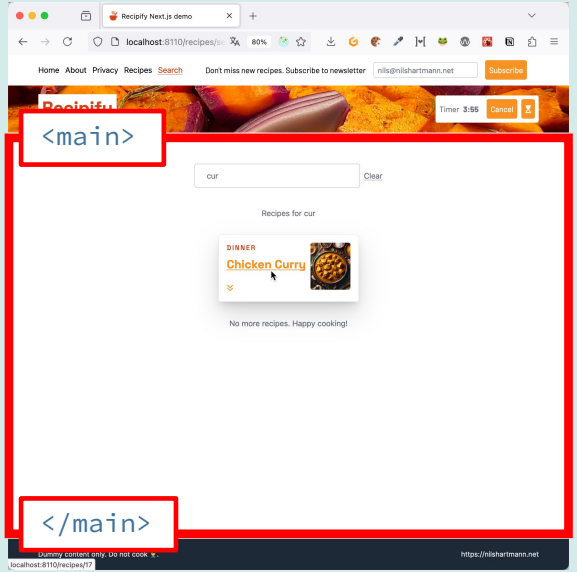
`RecipeSummaryCard.jte`



Beispiel: SPA-like Seitenwechsel in HTMX

Variante 1

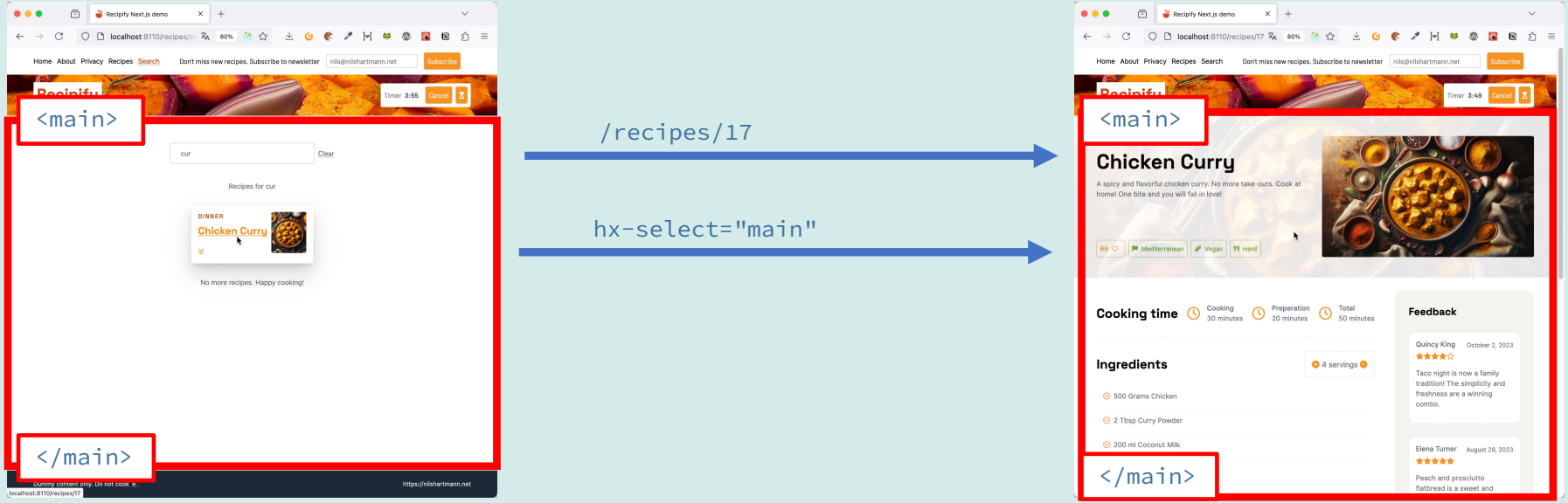
`RecipeSummaryCard.jte`



Beispiel: SPA-like Seitenwechsel in HTMX

Variante 1

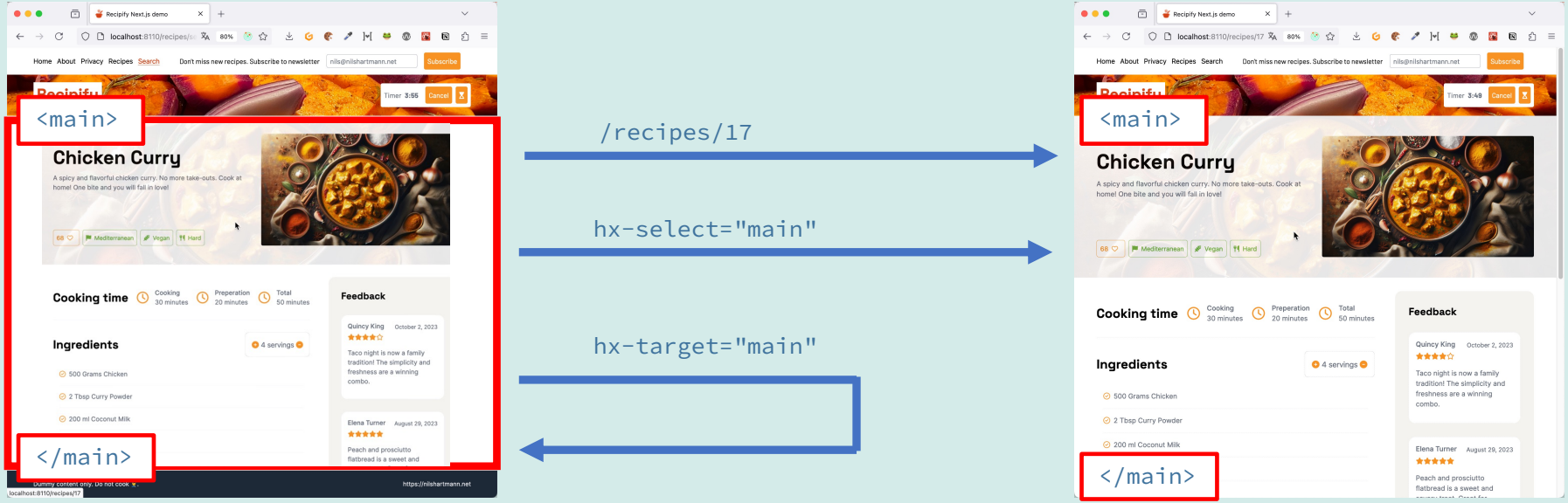
`RecipeSummaryCard.jte`



Beispiel: SPA-like Seitenwechsel in HTMX

Variante 1

`RecipeSummaryCard.jte`

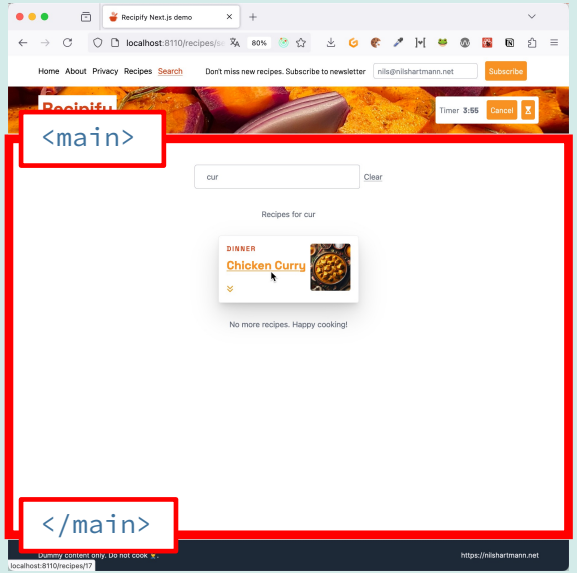


Variante 1: HTMX

- **Link** muss wissen:
  - wie sieht das Ergebnis aus
  - welches Element brauche ich daraus
  - wo muss das auf der aktuellen Seite eingebaut werden

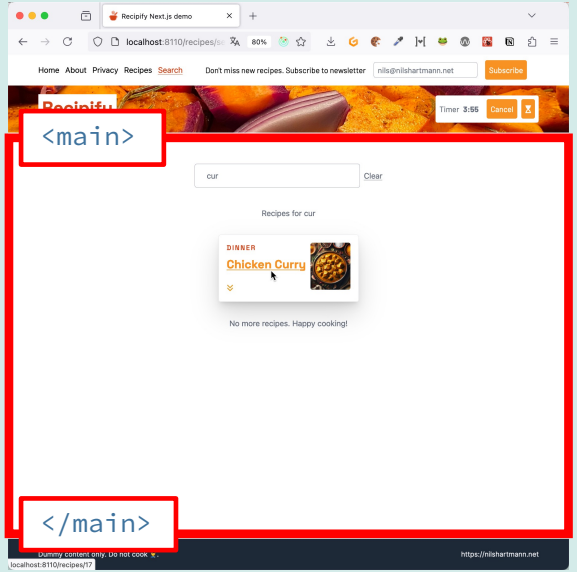
Beispiel: SPA-like Seitenwechsel in HTMX

Variante 2: Eigener Endpunkt

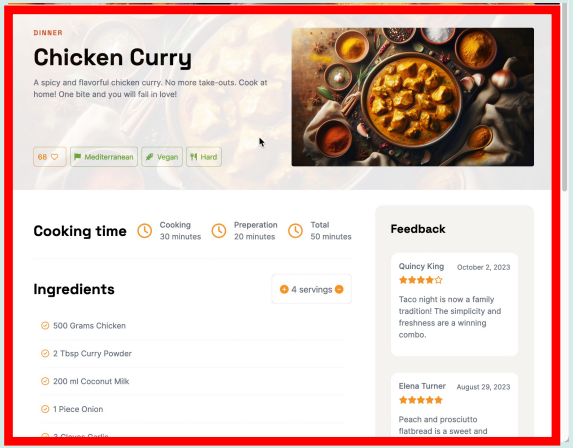


Beispiel: SPA-like Seitenwechsel in HTMX

Variante 2: Eigener Endpunkt

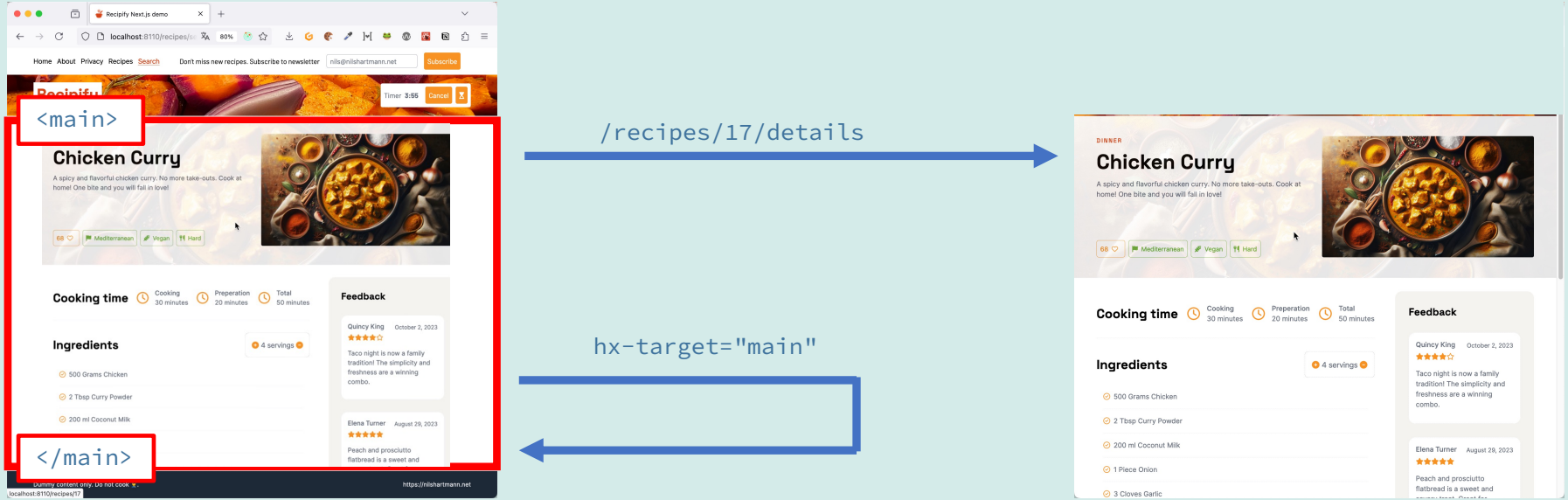


/recipes/17/details



Beispiel: SPA-like Seitenwechsel in HTMX

Variante 2



Variante 2: Eigener Endpunkt

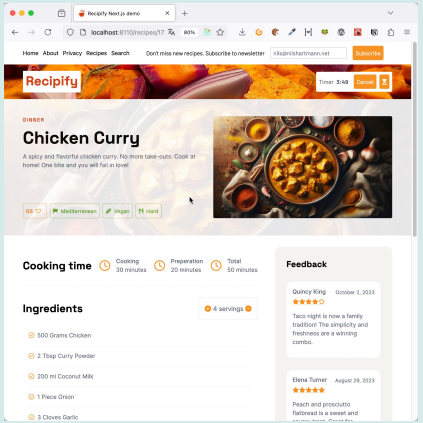
- **Link** muss wissen:
  - welcher Endpunkt
  - wo muss Ergebnis auf der aktuellen Seite eingebaut werden

Variante 2: Eigener Endpunkt

- **Backend** muss zwei Endpunkte zur Verfügung stellen:

Variante 2: Eigener Endpunkt

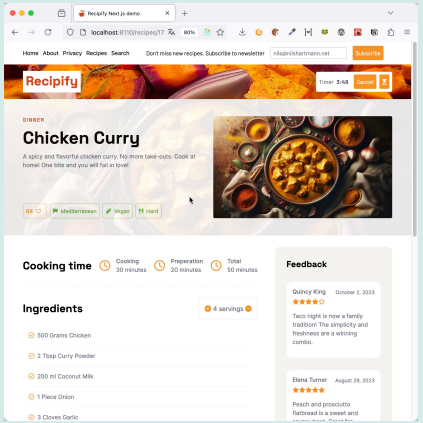
- **Backend** muss zwei Endpunkte zur Verfügung stellen:
  - Ganze Seite



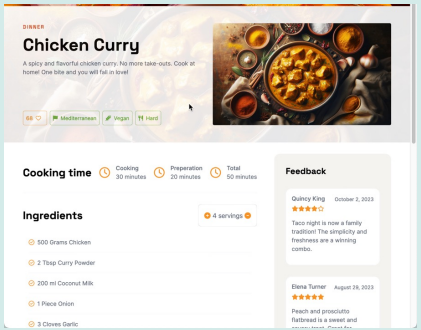
`/recipes/17`

Variante 2: Eigener Endpunkt

- **Backend** muss zwei Endpunkte zur Verfügung stellen:
  - Ganze Seite
  - Nur den Mittelteil



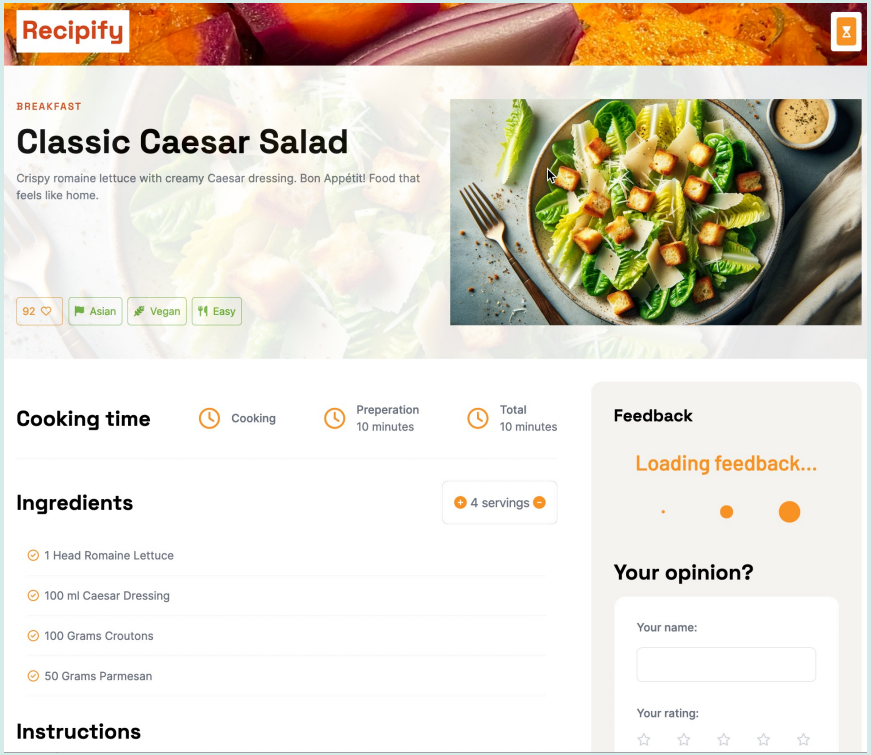
/recipes/17



/recipes/17/details

CODE-BEISPIELE

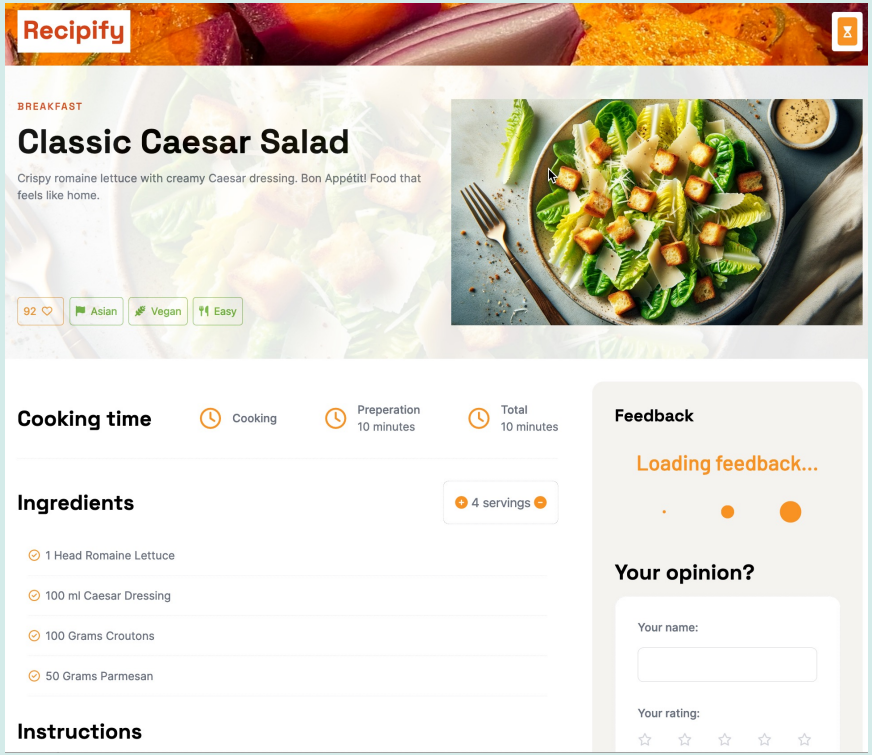
Beispiel: Seite priorisieren



CODE-BEISPIELE

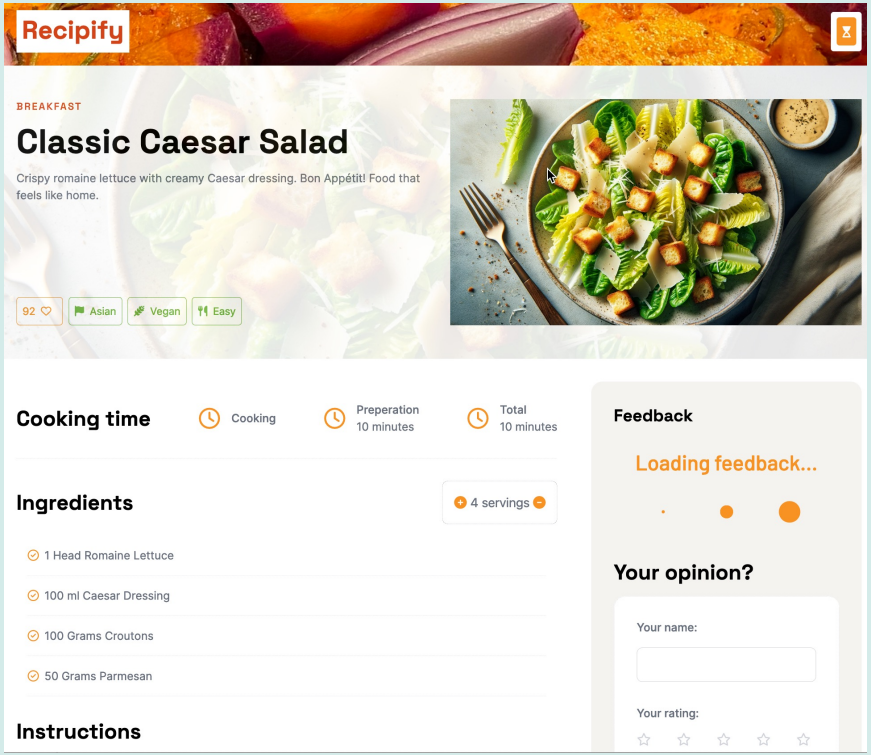
Beispiel: Seite priorisieren in Next.js

- 🕵️ `slowDown_GetFeedbacks=3000`
- 🕵️ `slowDown_GetRecipe=2000`
- 🕵️ `Spring Controller Debug`
- 🕵️ `Netzwerk-Tab Gesamtdauer`
- 🕵️ `[recipeId]/page.tsx`
- 🕵️ `RecipePageContent.tsx`



CODE-BEISPIELE

Beispiel: Seite priorisieren in HTMX



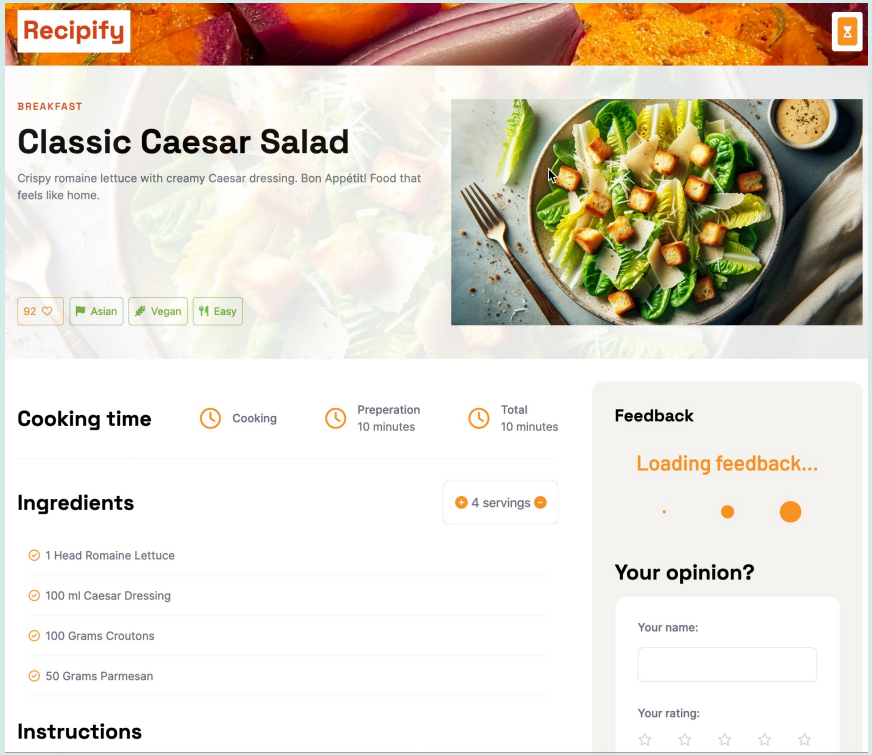
CODE-BEISPIELE

Beispiel: Seite priorisieren in HTMX

🕵️ `Feedback.jte`

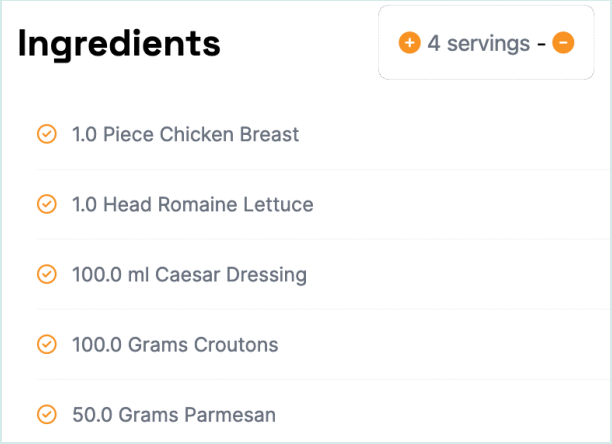
🕵️ `http://localhost:8080/recipes/25?slowdown=2000`

🕵️ `Netzwerk-Tab`



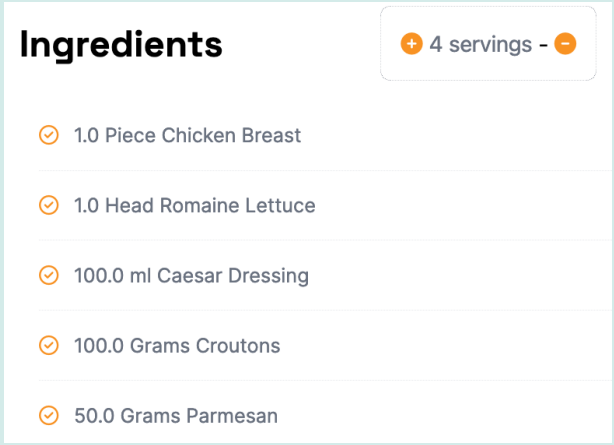
Beispiel: Clientseitige Interaktion

- Clientseitige Interaktion



Beispiel: Clientseitige Interaktion in Next.js

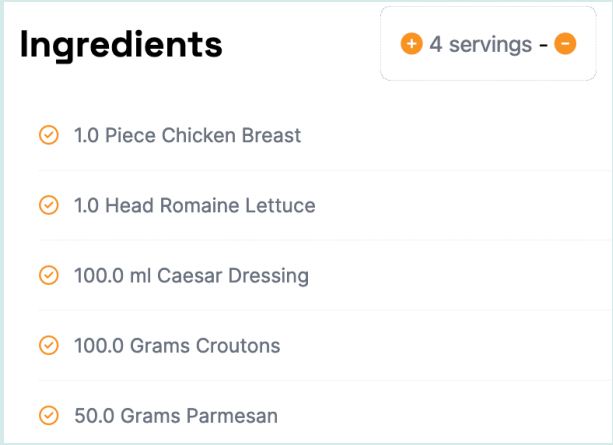
`IngredientsSection.tsx`



Beispiel: Clientseitige Interaktion in HTMX

- Wir brauchen spätestens hier JavaScript

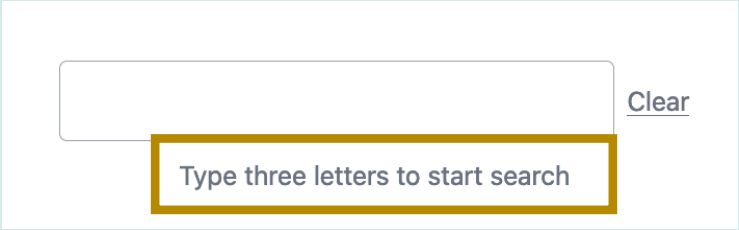
🕵️ `Ingredients.jte`



Beispiel: Clientseitige Interaktion in HTMX

- Wir brauchen spätestens hier JavaScript

🕵️ `SearchPageContent.jte`



Fazit

FAZIT: SINGLE-PAGE-ANWENDUNG ODER HTMX?

Single-Page-Anwendung oder HTMX

- It depends (natürlich 🙄)

FAZIT: SINGLE-PAGE-ANWENDUNG ODER HTMX?

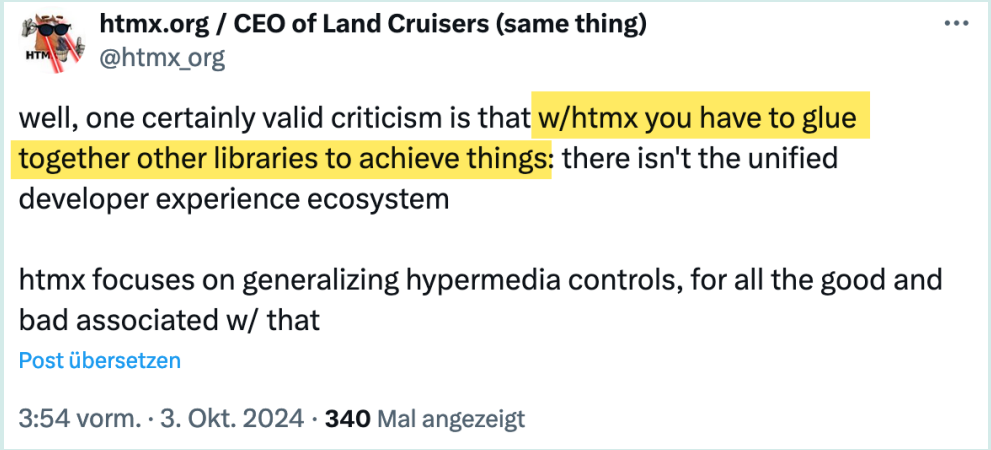
Single-Page-Anwendung oder HTMX

- Für mich ist "wir machen nur HTML plus bisschen JS" zu kurz gesprungen
- Das sieht man auch an der Vielzahl der HTMX-Attribute

FAZIT: SINGLE-PAGE-ANWENDUNG ODER HTMX?

Single-Page-Anwendung oder HTMX

- Für mich ist "wir machen nur HTML plus bisschen JS" zu kurz gesprungen
- Das sieht man auch an der Vielzahl der HTMX-Attribute
- Früher oder später braucht man doch JavaScript



https://x.com/htmx_org/status/1841658035984646318

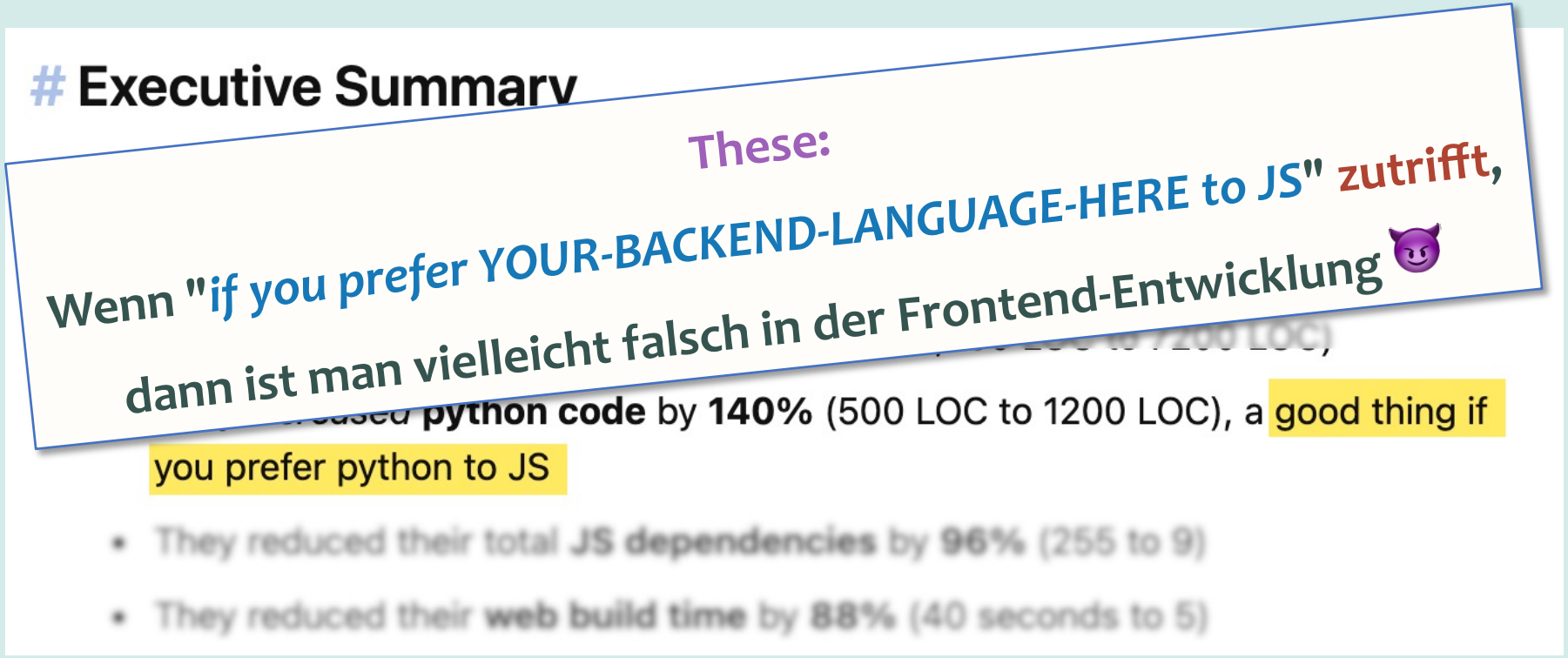
FAZIT: SINGLE-PAGE-ANWENDUNG ODER HTMX?

Executive Summary

- The effort took about **2 months** (with a 21K LOC code base, mostly JavaScript)
- **No reduction** in the application's user experience (UX)
- They reduced the **code base size** by **67%** (21,500 LOC to 7200 LOC)
- They *increased* **python code** by **140%** (500 LOC to 1200 LOC), a good thing if you prefer python to JS
- They reduced their total **JS dependencies** by **96%** (255 to 9)
- They reduced their **web build time** by **88%** (40 seconds to 5)

https://htmx.org/essays/a-real-world-react-to-htmx-port/

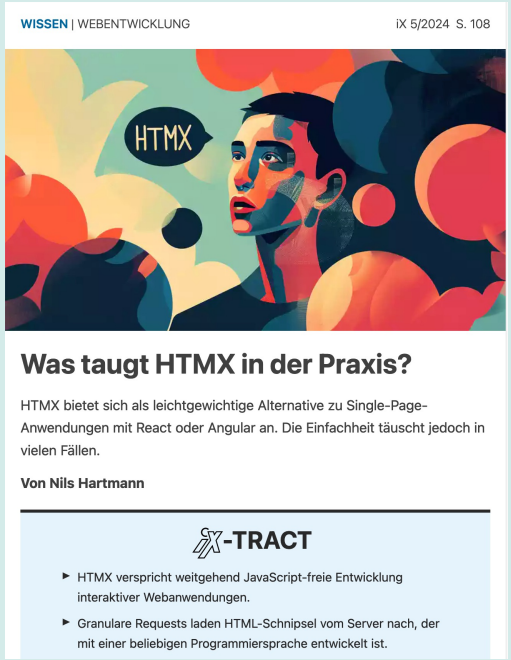
FAZIT: SINGLE-PAGE-ANWENDUNG ODER HTMX?



https://htmx.org/essays/a-real-world-react-to-htmx-port/

Mein Artikel (heise / iX) zum Thema:

https://react.schule/heise-htmx



WISSEN | WEBENTWICKLUNG iX 5/2024 S. 108

Was taugt HTMX in der Praxis?

HTMX bietet sich als leichtgewichtige Alternative zu Single-Page-Anwendungen mit React oder Angular an. Die Einfachheit täuscht jedoch in vielen Fällen.

Von Nils Hartmann

iX-TRACT

- HTMX verspricht weitgehend JavaScript-freie Entwicklung interaktiver Webanwendungen.
- Granulare Requests laden HTML-Schnipsel vom Server nach, der mit einer beliebigen Programmiersprache entwickelt ist.

DEMO

Beispiel-Anwendung

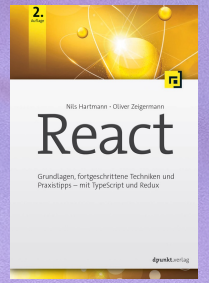
Source-Code und Slides:

`https://react.schule/wjax-2024-htmx`

Nach dem Vortrag checke ich den geschriebenen "Live-Code" auf einem neuen Branch ein

NILS HARTMANN
https://nilshartmann.net

https://reactbuch.de



Vielen Dank!

Slides & Code: https://react.schule/wjax-2024-htmx

Fragen und Kontakt

nils@nilshartmann.net

https://nilshartmann.net/kontakt