Laboratorio di "Sistemi Distribuiti"
A.A. 2024-2025

JavaScript (Fetch API)

Emanuele Petriglia

Slide aggiornate al 4 Giugno 2025

Indice dei contenuti

1. Introduzione

Fetch API

La **Fetch API** è una API Web che fornisce un'interfaccia per ottenere risorse remote tramite richieste HTTP asincrone. Le interfacce esposte sono quattro:

1. `fetch()`: una funzione globale che si utilizza per ottenere una risorsa effettuando una richiesta,
2. `Response`: un'interfaccia che rappresenta la risposta a una richiesta,
3. `Request`: un'interfaccia che rappresenta la richiesta da passare a `fetch()`,
4. `Headers`: rappresenta un insieme di header, sia in lettura (invio) sia in scrittura (ricezione).

Alcune differenze con `XMLHttpRequest`

- Fetch è stata introdotta nel 2015, `XMLHttpRequest` nel 2002.
- Fetch si integra con il modello di esecuzione asincrona di JavaScript, `XMLHttpRequest` no.
- Fetch integra il parsing del JSON, `XMLHttpRequest` no.
- Fetch espone API moderne, flessibili e più facili da usare rispetto a `XMLHttpRequest`.

Attenzione!

AJAX (**A**synchronous **J**avaScript **a**nd **X**ML) è un pattern di programmazione, non una tecnologia. Sia `XMLHttpRequest` che Fetch implementano il pattern.

Abilitare CORS

CORS

Per motivi di sicurezza, i browser impediscono a `fetch()` di fare richieste verso un dominio diverso da quello originale.

CORS ("Cross-Origin Resource Sharing") è un meccanismo di sicurezza regola l'ottenimento delle risorse da una pagina web ad un altro dominio.

CORS non viene trattato nel laboratorio, ma:

- Per l'hello world, è necessario scaricare e installare l'estensione **Allow CORS** per Google Chrome e abilitarlo (click sull'icona).
- Per gli esercizi (e progetto) il server Web che espone le API contiene le regole per permettere le chiamate `fetch()`. Non è necessaria alcuna estensione.

1. Introduzione

Modello asincrono di JavaScript

Modello asincrono di JavaScript (pt. 1)

- `fetch()` restituisce un oggetto di tipo `Promise`.
- `Promise` è un oggetto che rappresenta il risultato di un'operazione che viene eseguita in modo asincrono, cioè senza bloccare il flusso di esecuzione del programma.

Qualche indicazione

Con l'oggetto `Promise` si possono fare diverse cose. Con i metodi `then(myFunction)` e `catch(myFunction)` è possibile indicare delle funzioni che verranno eseguite quando l'oggetto sarà "risolto".

`Promise` non è trattato dal laboratorio. Maggiori informazioni su MDN Web Docs.

Modello asincrono di JavaScript (pt. 2)

JavaScript ha le parole chiavi `async` e `await` per gestire gli oggetti `Promise` in modo agevole:

- `async function` permette di definire funzioni che fanno uso di `await` all'interno.
- `await` è l'operatore utilizzato per aspettare la risoluzione di una `Promise` e ottenerne il risultato.

Qualche indicazione

Si suggerisce sempre di usare `async`/`await`, rende il codice più leggibile.

`async`/`await` non sono trattati nel laboratorio. Maggiori informazioni su MDN Web Docs.

Esempio Promise e async/await (pt. 1)

```
1   function resolveAfter2Seconds() {
2     return new Promise((resolve) => {
3       setTimeout(() => {
4         resolve('resolved');
5       }, 2000);2937
6     });
7   }
8
9   function normalFunction() {
10    console.log("normalFunction: INIZIO");
11    const resolve = resolveAfter2Seconds();
12    console.log("normalFunction: Promise creata!");
13    resolve.then((result) => {
14      console.log(`normalFunction: ${result}`);
15    });
16    console.log("normalFunction: FINE");
17  }
18
19  async function asyncFunction() {
20    console.log("asyncFunction: INIZIO");
21    const result = await resolveAfter2Seconds();
22    console.log("asyncFunction: Promise creata e aspettata!");
23    console.log(`asyncFunction: ${result}`);
24    console.log("normalFunction: FINE");
25  }
26
27  asyncFunction();
28  normalFunction();
```

Esempio Promise e async/await (pt. 2)

```
> "asyncFunction: INIZIO"
> "normalFunction: INIZIO"
> "normalFunction: Promise creata!"
> "normalFunction: FINE"
> "asyncFunction: Promise creata e aspettata!"
> "asyncFunction: resolved"
> "normalFunction: FINE"
> "normalFunction: resolved"
```

1. Introduzione

Il metodo `fetch()`

fetch()

`fetch()` è una funzione globale che accetta due parametri (vedremo dopo il secondo), di cui il primo è la risorsa da ottenere. Di base, senza opzioni, effettua una richiesta HTTP GET alla risorsa indicata.

Restituisce un oggetto di tipo `Promise`, è necessario quindi gestire il risultato con `then()`/`catch()` oppure ottenere il risultato con `await`. In entrambi i casi, il risultato in caso di successo è un oggetto di tipo `Response`.

Alcune proprietà di `Response`:

- `Response.ok`: un booleano che indica se la richiesta ha avuto successo,
- `Response.status`: codice di stato HTTP della risposta.
- `Response.json()`: effettua il parsing in JSON del body e restituisce una `Promise` che, una volta risolta, restituisce l'oggetto JavaScript.

Esempio: preparazione

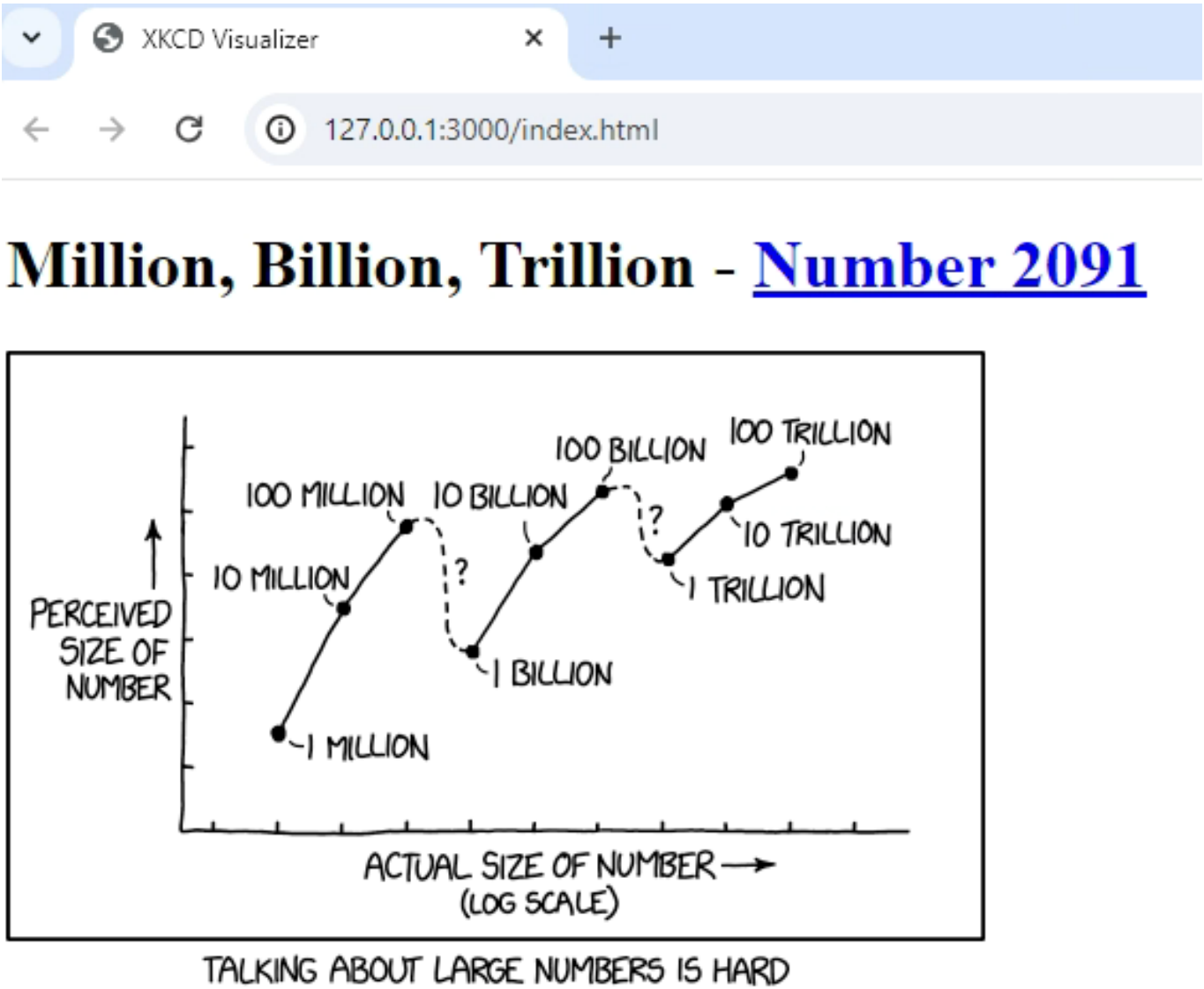
1. Scaricare, estrarre ed aprire il progetto "Hello World" presente su e-Learning con Visual Studio Code.
2. Abilitare la Live Preview.

Attenzione

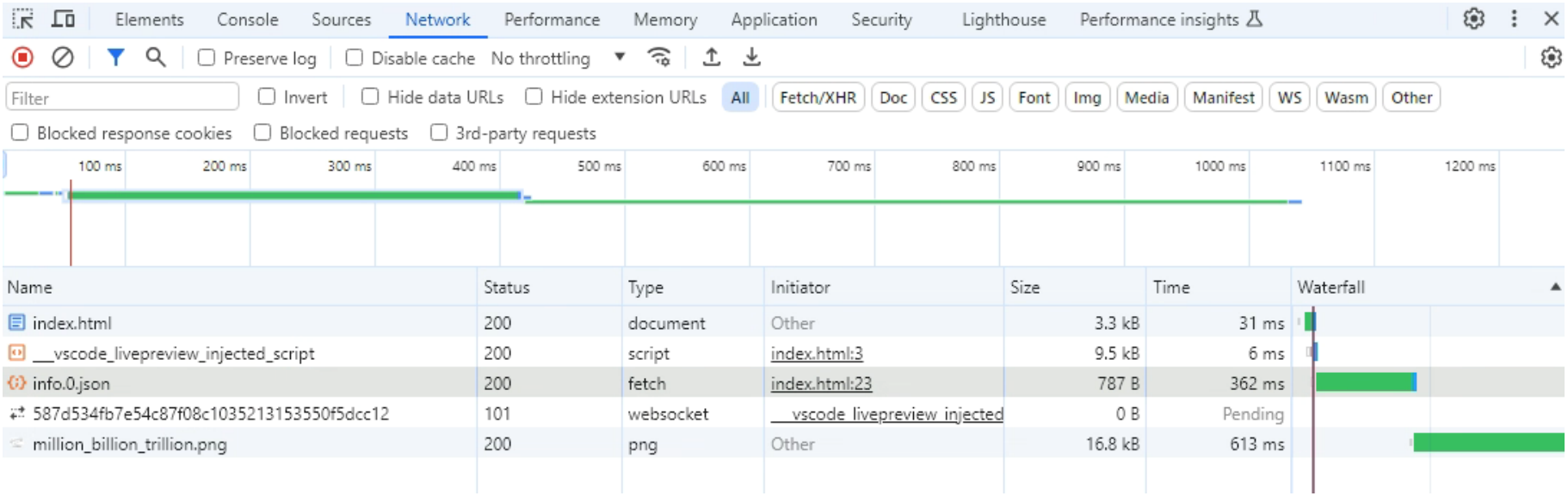
La Live Preview ha CORS non configurato. È necessario aprire la pagina con Chrome!

3. Aprire Chrome e visualizzare il risultato, ricaricare più volte la pagina.
4. Visualizzare le schede **Elements**, **Console** e **Network** negli strumenti di sviluppo.

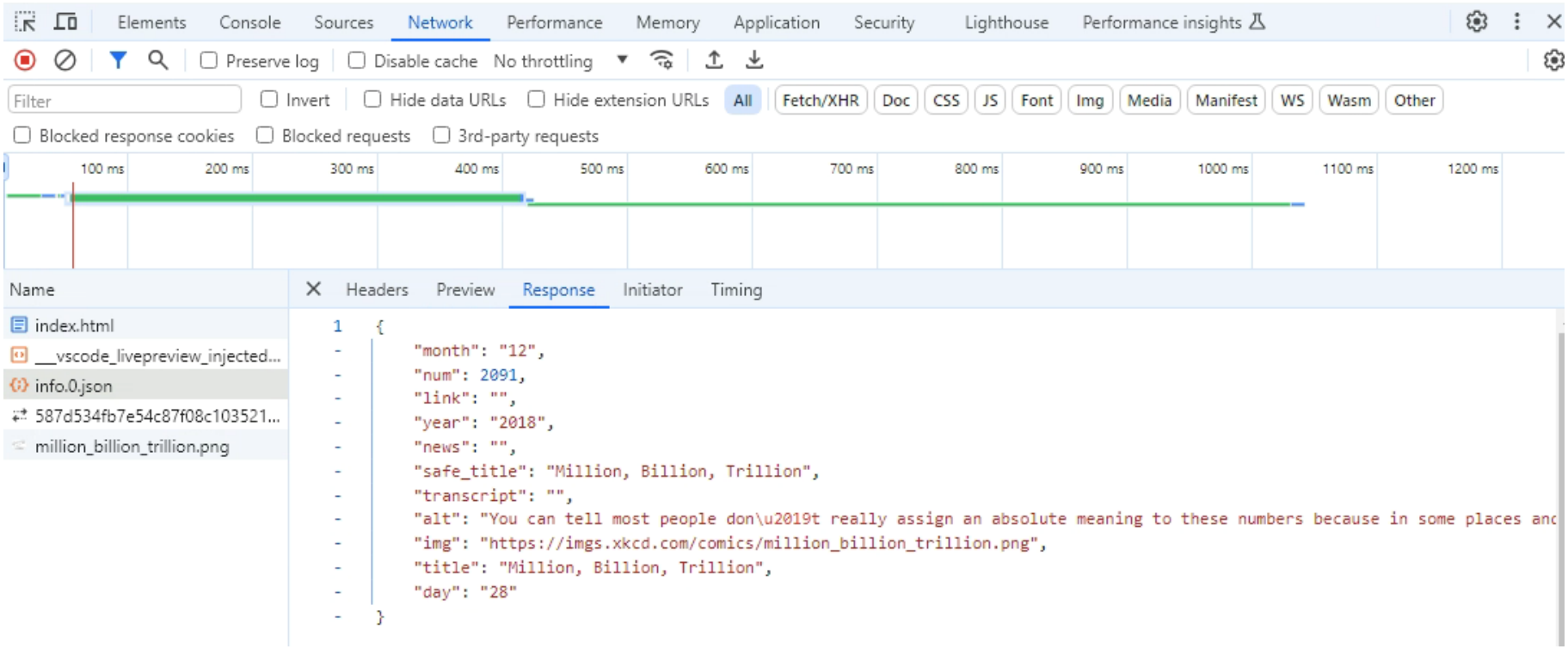
Esempio: risultato (pt. 1)



Esempio: risultato (pt. 2)



Esempio: risultato (pt. 3)



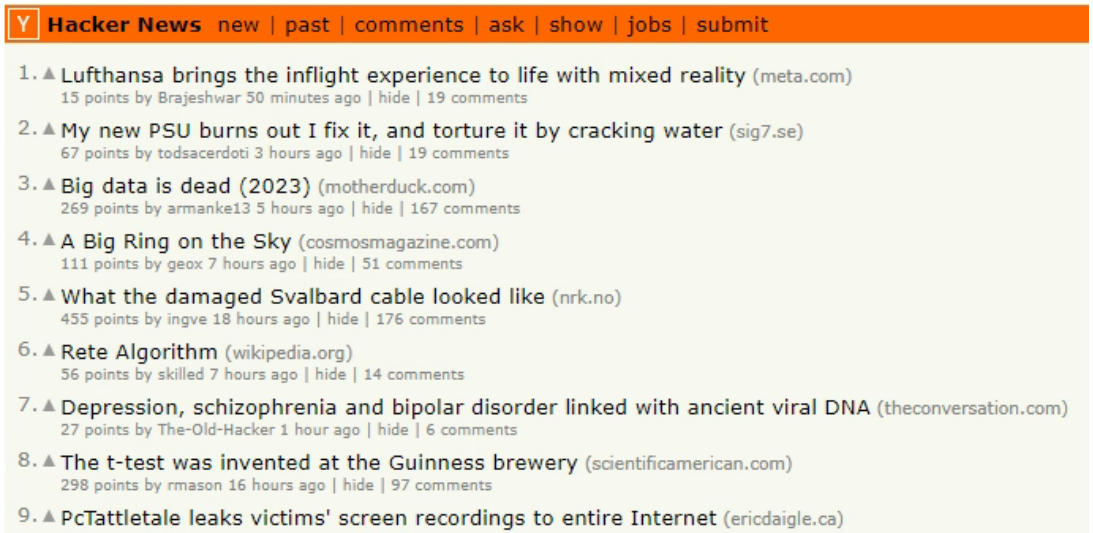
Esempio: codice

```
1  async function getRandomComic() {
2      const latest_comic_number = 2937;
3      const comicId = Math.floor(Math.abs(Math.random())
       ↪  * latest_comic_number);
4
5      const response = await
       ↪  fetch(`https://xkcd.com/${comicId}/info.0.json`);
6
7      if (!response.ok) {
8          console.error("Request failed!");
9          return;
10     }
11     return await response.json();
12 }
13
14 const comic = await getRandomComic();
```

2. Esercizio A: Chiamate GET con `fetch`

Consegna (pt. 1)

Ricreare una versione semplificata di **Hacker News** utilizzando l'API pubblica ufficiale.

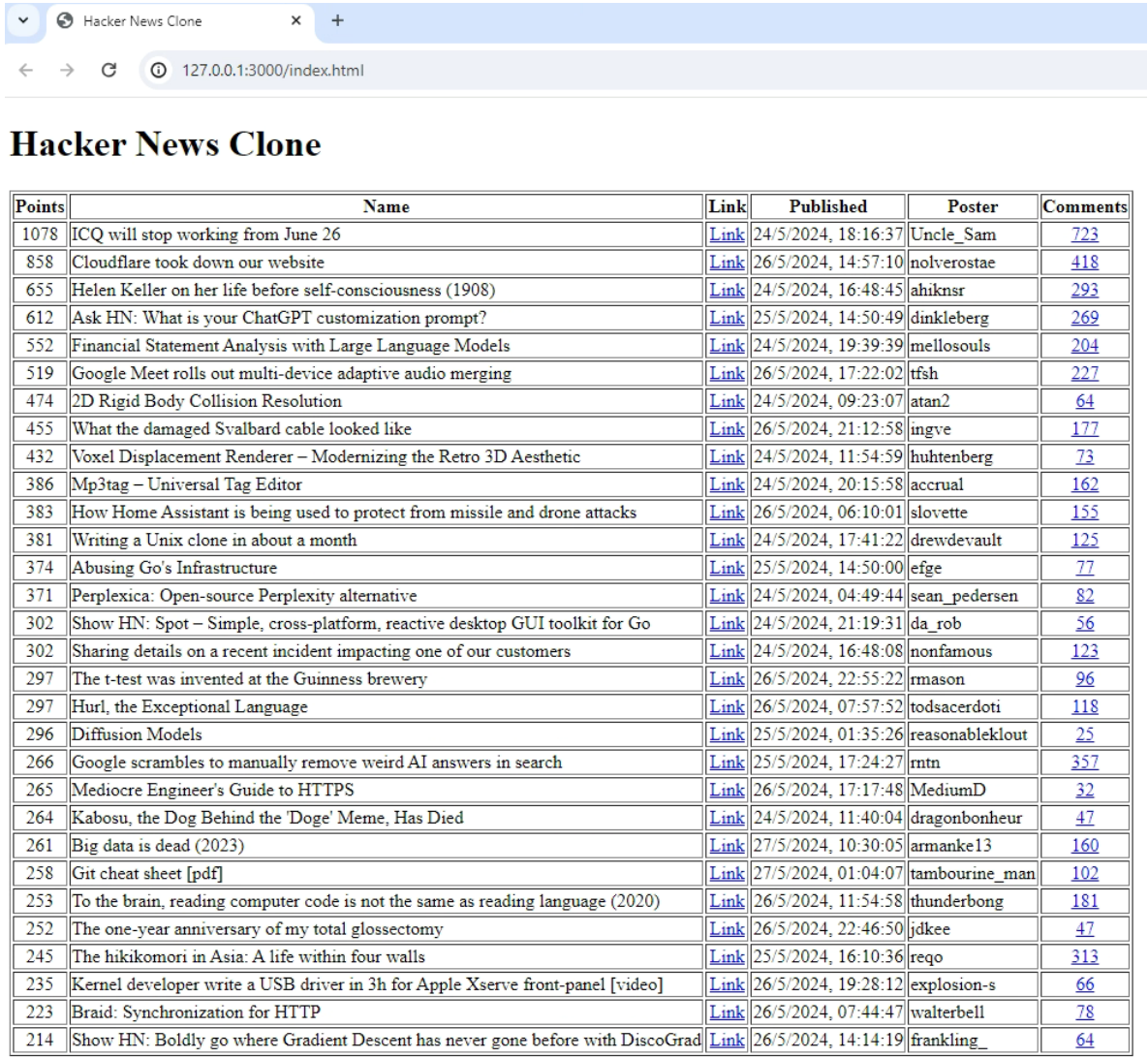


Al caricamento della pagina mostrare una tabella con le prime 30 storie della categoria "best" e per ognuna: il punteggio, il titolo, link alla notizia originale, data di pubblicazione, autore della storia e numero di commenti (con un link al sito ufficiale).

Partire dallo scheletro su e-Learning.

Consegna (pt. 2)

Il risultato deve essere uguale al seguente (il contenuto può variare):



Points	Name	Link	Published	Poster	Comments
1078	ICQ will stop working from June 26	Link	24/5/2024, 18:16:37	Uncle_Sam	723
858	Cloudflare took down our website	Link	26/5/2024, 14:57:10	nolverostae	418
655	Helen Keller on her life before self-consciousness (1908)	Link	24/5/2024, 16:48:45	ahiknsr	293
612	Ask HN: What is your ChatGPT customization prompt?	Link	25/5/2024, 14:50:49	dinkleberg	269
552	Financial Statement Analysis with Large Language Models	Link	24/5/2024, 19:39:39	mellosouls	204
519	Google Meet rolls out multi-device adaptive audio merging	Link	26/5/2024, 17:22:02	tfsh	227
474	2D Rigid Body Collision Resolution	Link	24/5/2024, 09:23:07	atan2	64
455	What the damaged Svalbard cable looked like	Link	26/5/2024, 21:12:58	ingve	177
432	Voxel Displacement Renderer – Modernizing the Retro 3D Aesthetic	Link	24/5/2024, 11:54:59	huhtenberg	73
386	Mp3tag – Universal Tag Editor	Link	24/5/2024, 20:15:58	accrual	162
383	How Home Assistant is being used to protect from missile and drone attacks	Link	26/5/2024, 06:10:01	slovette	155
381	Writing a Unix clone in about a month	Link	24/5/2024, 17:41:22	drewdevault	125
374	Abusing Go's Infrastructure	Link	25/5/2024, 14:50:00	efge	77
371	Perplexica: Open-source Perplexity alternative	Link	24/5/2024, 04:49:44	sean_pedersen	82
302	Show HN: Spot – Simple, cross-platform, reactive desktop GUI toolkit for Go	Link	24/5/2024, 21:19:31	da_rob	56
302	Sharing details on a recent incident impacting one of our customers	Link	24/5/2024, 16:48:08	nonfamous	123
297	The t-test was invented at the Guinness brewery	Link	26/5/2024, 22:55:22	rmason	96
297	Hurl, the Exceptional Language	Link	26/5/2024, 07:57:52	todsacerdoti	118
296	Diffusion Models	Link	25/5/2024, 01:35:26	reasonableklout	25
266	Google scrambles to manually remove weird AI answers in search	Link	25/5/2024, 17:24:27	rntn	357
265	Mediocre Engineer's Guide to HTTPS	Link	26/5/2024, 17:17:48	MediumD	32
264	Kabosu, the Dog Behind the 'Doge' Meme, Has Died	Link	24/5/2024, 11:40:04	dragonbonheur	47
261	Big data is dead (2023)	Link	27/5/2024, 10:30:05	armanke13	160
258	Git cheat sheet [pdf]	Link	27/5/2024, 01:04:07	tambourine_man	102
253	To the brain, reading computer code is not the same as reading language (2020)	Link	26/5/2024, 11:54:58	thunderbong	181
252	The one-year anniversary of my total glossectomy	Link	26/5/2024, 22:46:50	jdkee	47
245	The hikikomori in Asia: A life within four walls	Link	25/5/2024, 16:10:36	reqo	313
235	Kernel developer write a USB driver in 3h for Apple Xserve front-panel [video]	Link	26/5/2024, 19:28:12	explosion-s	66
223	Braid: Synchronization for HTTP	Link	26/5/2024, 07:44:47	walterbell	78
214	Show HN: Boldly go where Gradient Descent has never gone before with DiscoGrad	Link	26/5/2024, 14:14:19	frankling_	64

Suggerimenti (pt. 1)

- Primo passo: consultare la **documentazione dell'API**, in particolare gli endpoint:
  - `/v0/item/<id>` per ottenere info su una storia,
  - `/v0/beststories` per ottenere l'elenco delle storie della categoria "best".

  Attenzione al suffisso finale "`.json`"!

- Come approcciare l'esercizio:
  1. Ottenere l'elenco delle storie nella categoria giusta con una `fetch()`,
  2. Ottenere le informazioni di ogni storia delle prime 30 dall'elenco (30 chiamate `fetch()`),
  3. Per ogni storia aggiungere una riga alla tabella con le informazioni ottenute (manipolazione DOM).

- Per fare pratica con `fetch()` è meglio stampare con `console.log()` ciò che succede man mano, tra cui quello che restituisce `response.json()`.

Suggerimenti (pt. 2)

- Il campo `time` di `item` restituito dall'API è definito in secondi, mentre in JavaScript in millisecondi. La conversione giusta è:

```
const date = new Date(story["time"] *
↪  1000).toLocaleString("it-IT");
```

- Al numero dei commenti per una storia bisogna associare il seguente link:

  `https://news.ycombinator.com/item?id=<id>`

- I template literals facilitano la creazione dell'URL in cui fare la richiesta con `fetch()`:

```
const url = `https://news.ycombinator.com/item?id=${story["id"]}`
```

3. Esercizio B: Altri metodi HTTP

Il parametro `options`

`fetch` accetta anche il parametro `options`: un oggetto che contiene le opzioni da usare per la richiesta.

Le opzioni più importanti sono:

- `method`: il metodo della richiesta (GET è predefinito, POST, DELETE, PUT...).
- `headers`: gli header da aggiungere alla richiesta.
- `body`: il corpo della richiesta

Esempio di POST con `fetch`

```
async function postFetch(object) {
    const endpoint = "http://example.com/collection";
    const response = await fetch(endpoint, {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify(object)
    });

    if (!response.ok) {
        console.error("Richiesta fallita!");
        return;
    }
    return response.headers.get("Location");
}
```

Consegna (pt. 1)

Realizzare un client Web per la gestione di utenti, tramite le API REST esposte da un server Java.

Il client Web deve permettere di visualizzare tutti gli utenti, rimuovere, aggiungere un nuovo utente o aggiornare il nome di uno.

Partire dallo scheletro presente su e-Learning. Contiene due cartelle:

- `server`: un server Web in Java che espone le API, da avviare con `mvn jetty:run` in un terminale,
- `client`: contiene il file `index.html` che è lo scheletro di partenza, contiene già il codice JavaScript per la gestione del DOM, **bisogna solo implementare** la parte delle chiamate `fetch()`.

Consegna (pt. 2)

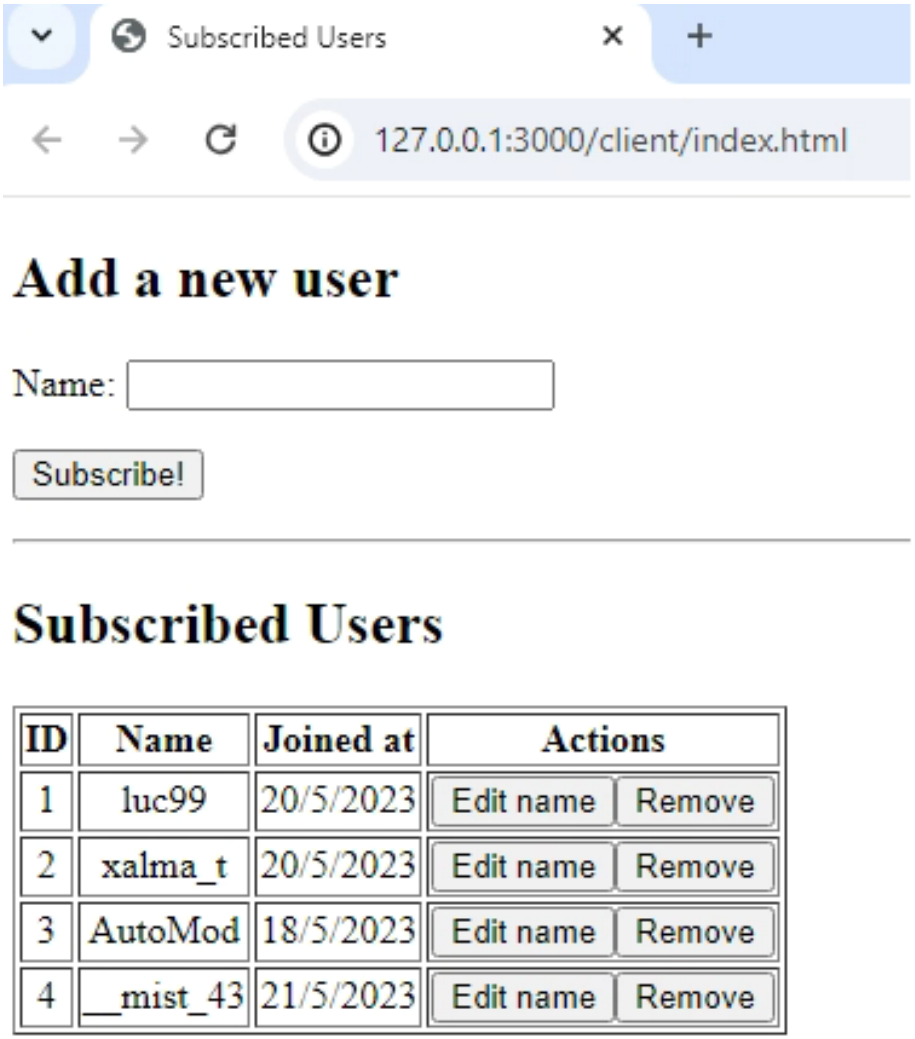
Le API esposte dal server sono:

Metodo	**URL**	**Descrizione**
GET	`/users`	Restituisce l'elenco degli utenti.
POST	`/users`	Aggiunge un nuovo utente.
GET	`/users/{id}`	Restituisce le info di un utente.
PUT	`/users/{id}/name`	Aggiorna il nome dell'utente.
DELETE	`/users/{id}`	Elimina uno specifico utente.

Ogni utente ha un ID pubblico, un nome e una data di iscrizione. Solo il nome è modificabile.

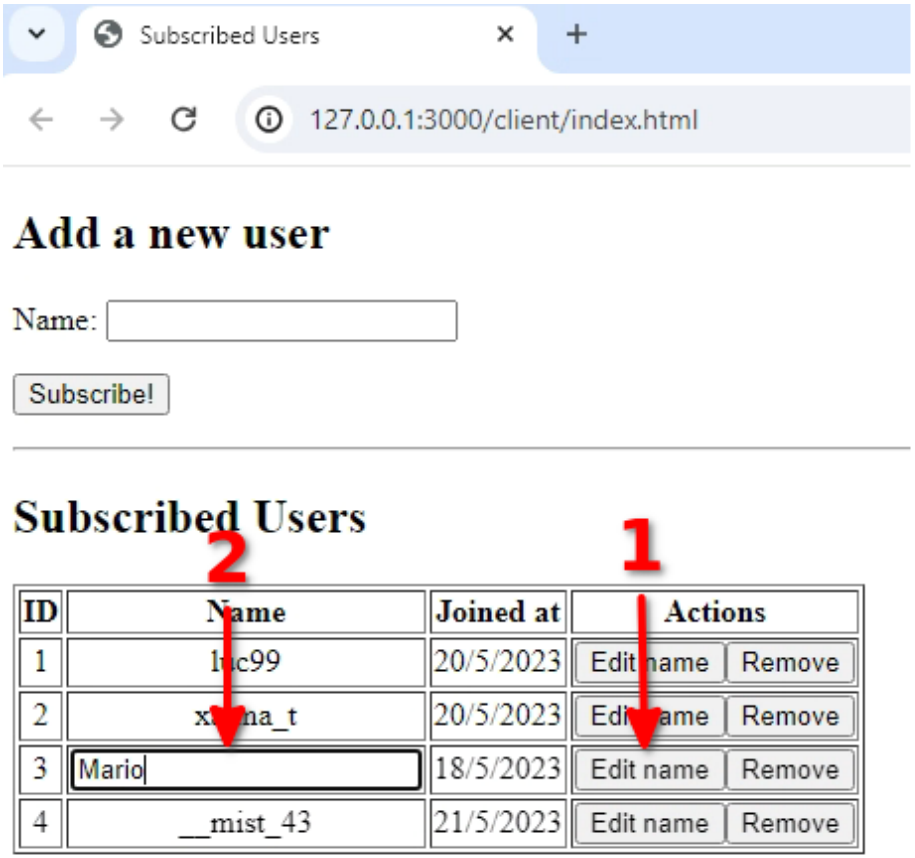
Consegna (pt. 3)

Il risultato deve essere simile al seguente:



Consegna (pt. 4)

Modifica del nome (bisogna premere invio per confermare):



4. Esercizio C: Back-end e front-end

Consegna (pt. 1)

Realizzare il front-end dell'API REST esposta nell'esercizio B del laboratorio 6 su REST.

Il client Web deve permettere di svolgere le operazioni di aggiunta/modifica/rimozione di una tastiera (come negli esercizi sulle Servlet).

Partire dallo scheletro presente su e-Learning. Contiene due cartelle:

- `server`: il server Web che espone le API, è la soluzione dell'esercizio B del lab. su REST (si può usare anche la propria soluzione).
- `client`: contiene il file `index.html` che è uno scheletro di partenza del client Web. Bisogna **implementare sia la parte di DOM sia le chiamate** `fetch()`.

Consegna (pt. 2)

Le API esposte dal server sono:

Metodo	**URL**	**Descrizione**
GET	`/keyboards`	Restituisce l'elenco delle tastiere.
POST	`/keyboards`	Aggiunge una nuova tastiera.
GET	`/keyboards/{id}`	Restituisce una specifica tastiera.
DELETE	`/keyboards/{id}`	Rimuove una specifica tastiera.
GET	`/keyboards/{id}/comments/{comment_id}`	Restituisce un singolo commento.
DELETE	`/keyboards/{id}/comments/{comment_id}`	Rimuove un commento specifico.
POST	`/keyboards/{id}/comments/`	Aggiunge un nuovo commento.
PUT	`/keyboards/{id}/{property}`	Aggiorna una specifica proprietà della tastiera.
GET	`/keyboards/{id}/{property}`	Restituisce una specifica proprietà della tastiera (tra cui i commenti).

Ogni tastiera ha le proprietà: `id`, `name`, `manufacturer`, `year`, `ergonomic`, `backlight`, `comments` e `layout`.

Consegna (pt. 3)

Il client Web deve permettere di:

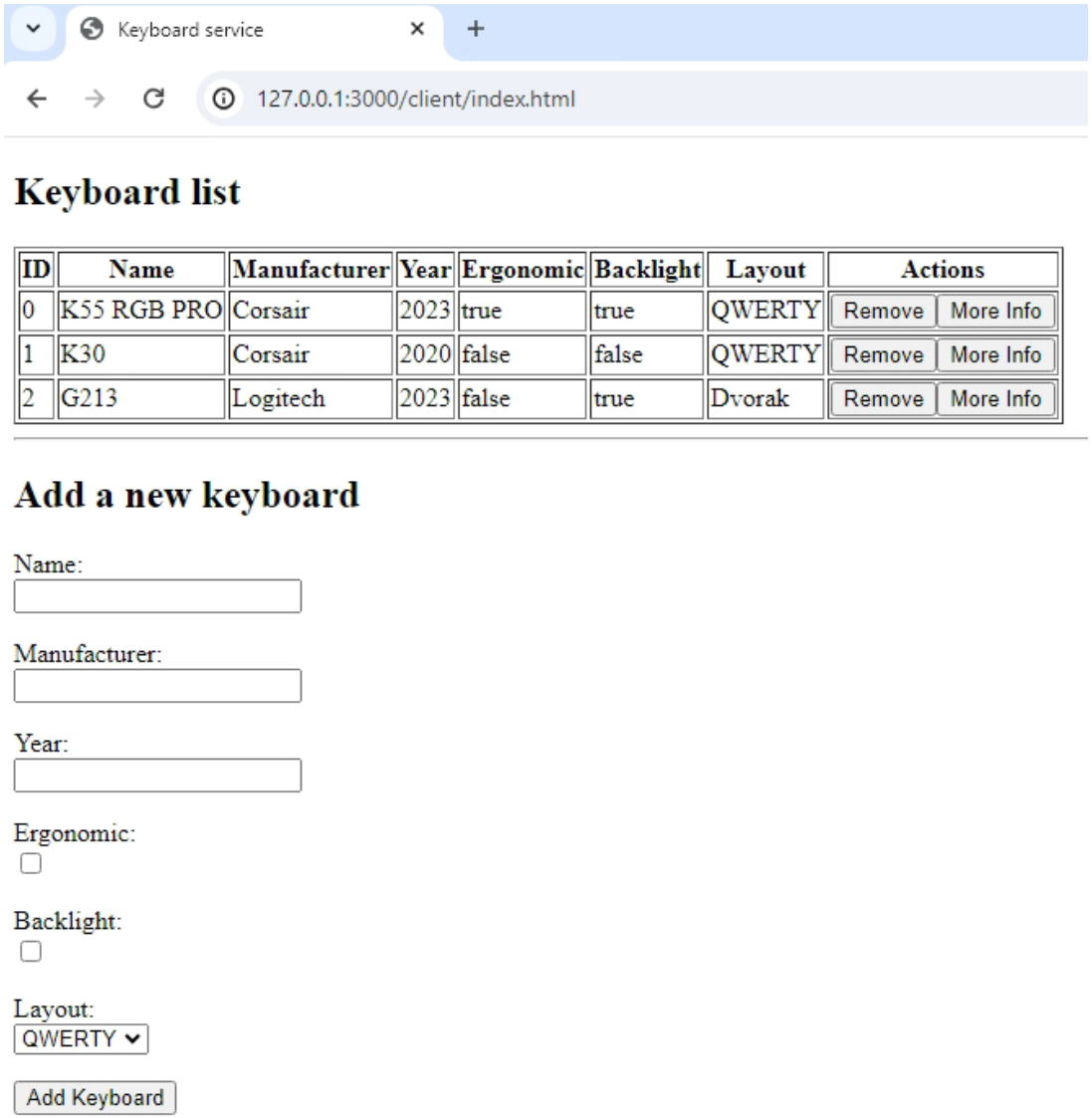
1. Visualizzare l'elenco delle tastiere,
2. Visualizzare i dettagli di una tastiera,
3. Aggiungere o rimuovere una tastiera,
4. Aggiungere o rimuovere un commento a una tastiera,

Attenzione

Non è necessario che il client utilizzi tutte le API esposte dal server!

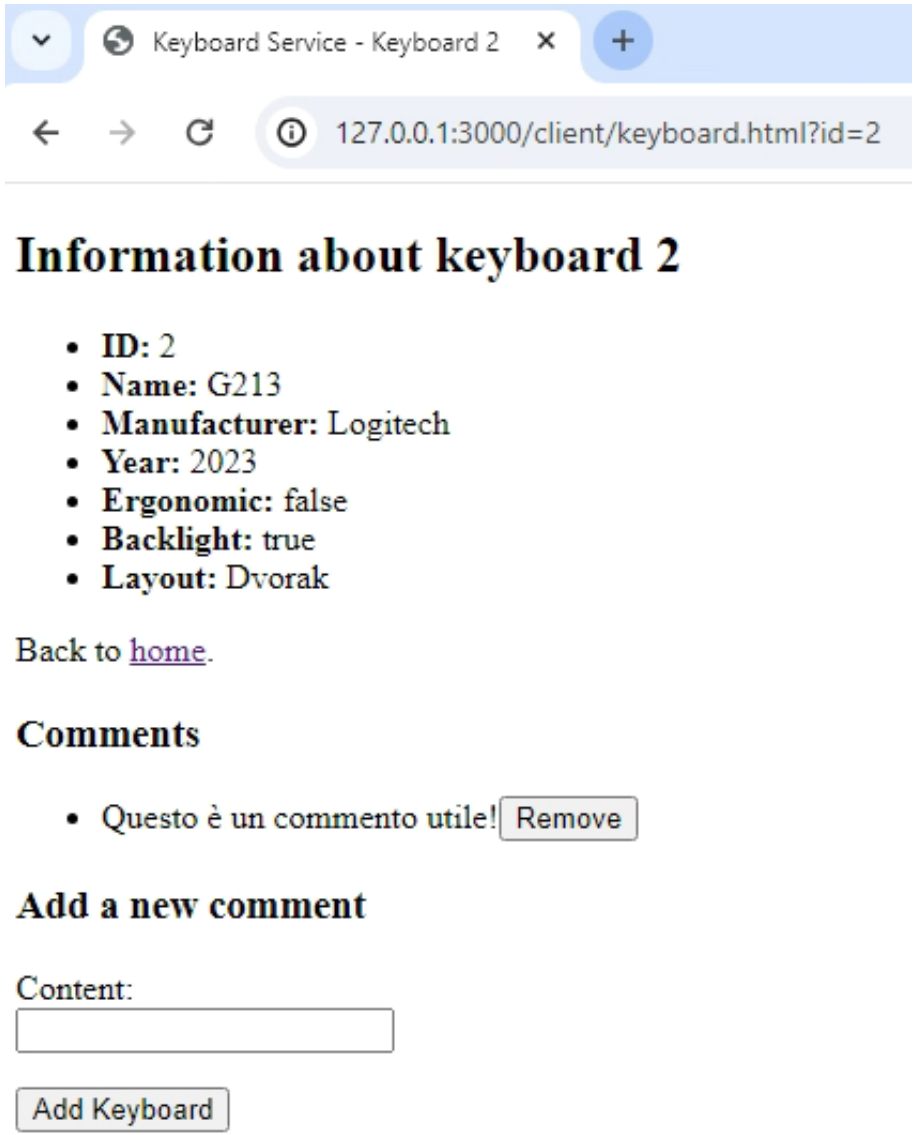
Consegna (pt. 4)

Un esempio di possibile soluzione è (pagina principale):



Consegna (pt. 5)

Un esempio di possibile soluzione è (pagina di una tastiera):



Suggerimenti

- È possibile creare più di una pagina Web, con all'interno codici JavaScript diversi racchiusi nell'elemento <script>.

  Oppure all'interno della pagina si può richiamare un file JavaScript nel seguente modo:

  ```
  <script src="nomeFile.js"></script>
  ```

- Per cambiare pagina si può fare nel seguente modo (pulsante che reagisce al click e che passa argomenti):

  ```
  1  btn.addEventListener("click", () => {
  2    window.location.href = `pagina.html?id=${id}`
  3  });
  ```

  Nella pagina `pagina.html` si può accedere agli argomenti di query con:

  ```
  1  const params = new
     ↪  URLSearchParams(document.location.search);
  2  const id = params.get("id");
  ```