

**SUPSI**

# Progetto pratico: Sviluppo client http con framework Svelte

---

Students

Alessandro Cantoni  
Mattia Verdolin

Course

Applicazioni web 2

Date

08.04.2025

# Indice

1. Introduzione
2. Components
3. Context
4. Demo presentation
5. Conclusioni



# Introduction

# Context and motivation – Project Overview

L'obiettivo principale è realizzare un componente configurabile in grado di:

- **Eseguire richieste HTTP (GET, POST, PUT, DELETE)**
- **Gestire headers e body personalizzati**
- **Visualizzare le risposte (HTML, immagini, PDF)**
- **Organizzare richieste in collections gestibili via sidebar**
- **Offrire funzionalità di ricerca, salvataggio, import/export**
- **Il componente dev'essere configurabile tramite parametri**

Viene utilizzato il framework Svelte con Vite.js per lo sviluppo rapido e modulare.

## Context and motivation – Svelte Overview

Svelte è un moderno framework JavaScript per lo sviluppo di interfacce utente reattive, noto per la sua semplicità e prestazioni elevate.

A differenza di altri framework come React o Vue, Svelte non utilizza un virtual DOM, ma compila i componenti in JavaScript ottimizzato al momento della build.



## Context and motivation – Svelte Overview

Questo si traduce in:

- Applicazioni più veloci e leggere
- Meno codice boilerplate
- Maggiore controllo sul comportamento dei componenti

Nel nostro progetto, Svelte ha facilitato la comunicazione tra componenti, la gestione dello stato e la creazione di un'interfaccia dinamica e modulare.





## Components

# Divisione in componenti – big division

The screenshot displays a web client interface with a sidebar on the left and a main content area on the right. The sidebar, titled 'Collections', contains a search bar and a list of collections: 'Collection 1' (selected), 'Request 1', and 'Collection 2'. The main content area shows a POST request to 'http://google.com' with a 'Content-Type' header of 'application/json'. The response is a 200 OK status with a 523 ms response time and 8.74 KB of data. The response body is an HTML document from Google, starting with the DOCTYPE declaration and the HTML element, followed by the head section containing meta tags for content type and branding, and a script tag for Google Analytics.

**Collections**

Search

Collection 1

Request 1

Collection 2

POST http://google.com Send

Header	Value
Content-Type	application/json

Request body content

200 OK 523 ms 8.74 KB Preview

```
<!doctype html>
<html itemscope="" itemtype="http://schema.org/WebPage" lang="de-CH">

<head>
  <meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
  <meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image">
  <title>Google</title>
  <script nonce="Gv_bEXRqVH-ZOkKuhZ59UQ">
    (function(){var _g=
    {kEl:'msS0Z9iJEseGvr0Pzdjc2QU',kEXPI:'0,4242917,2891,89155,344796,212051,47082,30911,5241523,16273045,20539756,25228681,78017,34107,94,10
    885,15165,8182,10492,38937,21670,6754,23879,9138,3078,1522,328,6225,1116,9302,23893,29145,709,1346,5323,8379,15635,21302,33,9041,17667,106
    65,5,14561,6780,2986,1501,13,4,3836,41,6133,7506,1,4914,1827,2847,1259,303,4148,16656,3074,2,919,718,232,928,1,2,1046,170,2833,1784,5774,4310,
    2064,262,1310,738,292,4087,1857,1731,143,10817,820,928,2,1920,535,12319,3261,43,416,1638,893,35,942,1317,6,1370,499,639,2,1127,1275,55,1027,1,
    142,66,330,2,288,218,1,2637,1678,1546,2,781,953,206,363,1871,1699,1046,870,465,9,369,1122,65,49,1224,523,2,1231,229,209,201,354,305,353,386,139,
    3,2755,3,466,720,537,5,102,8,109,3336,857,334,15,360,350,687,3,2,2,2,2,2,985,192,265,527,751,601,1036,415,478,9,1138,1882,606,220,801,210,253,29
    4,448,903,9,527,108,408,117,192,1,99,238,87,228,2177,160,893,331,74,390,45,13,685,20,7,59,402,228,936,966,949,412,223,364,17,4,172,772,49,97,6,11,
    529,235,1,3,5,3,3,2,516,206,109,12,722,410,251,21,152,24,84,3,2,6,1215,327,1806,179,167,147,31,77,891,840,20948562,375837,4,28852,18,2010,43,1171
    1316 8 3873 12 919' kRI '-LHn' kOPI:89978449):(function(){var a=(a=window.google)==null?0:a.stvs()&google.kFI= a.kFI>window.google= a})();call(this)})(/
```



# Divisione in componenti – big division



# Divisione in componenti – small division

The screenshot shows a REST client interface with the following components:

- Collections:** A sidebar on the left with a search bar and a list of collections: "Collection 1" (selected) and "Collection 2". "Request 1" is listed under "Collection 1".
- Request Method and URL:** A dropdown menu set to "POST" and a text input field containing "http://google.com".
- Send Button:** A green button labeled "Send" in the top right corner.
- Headers Table:** A table with two columns: "Header" and "Value". It contains one row: "Content-Type" with the value "application/json". There are three small "x" icons to the right of the table.
- Request Body Content:** A large text area labeled "Request body content" containing a large block of JSON data.
- Response Status and Size:** A row of three buttons: "200 OK", "523 ms", and "8.74 KB".
- Preview Button:** A green button labeled "Preview" in the bottom right corner.
- Response Preview:** A large text area at the bottom showing the HTML response from Google, including the doctype, meta tags, and a large block of JSON data.

# Divisione in componenti - composizione

Main component

Macro-components

Atomic components



# ③ Context

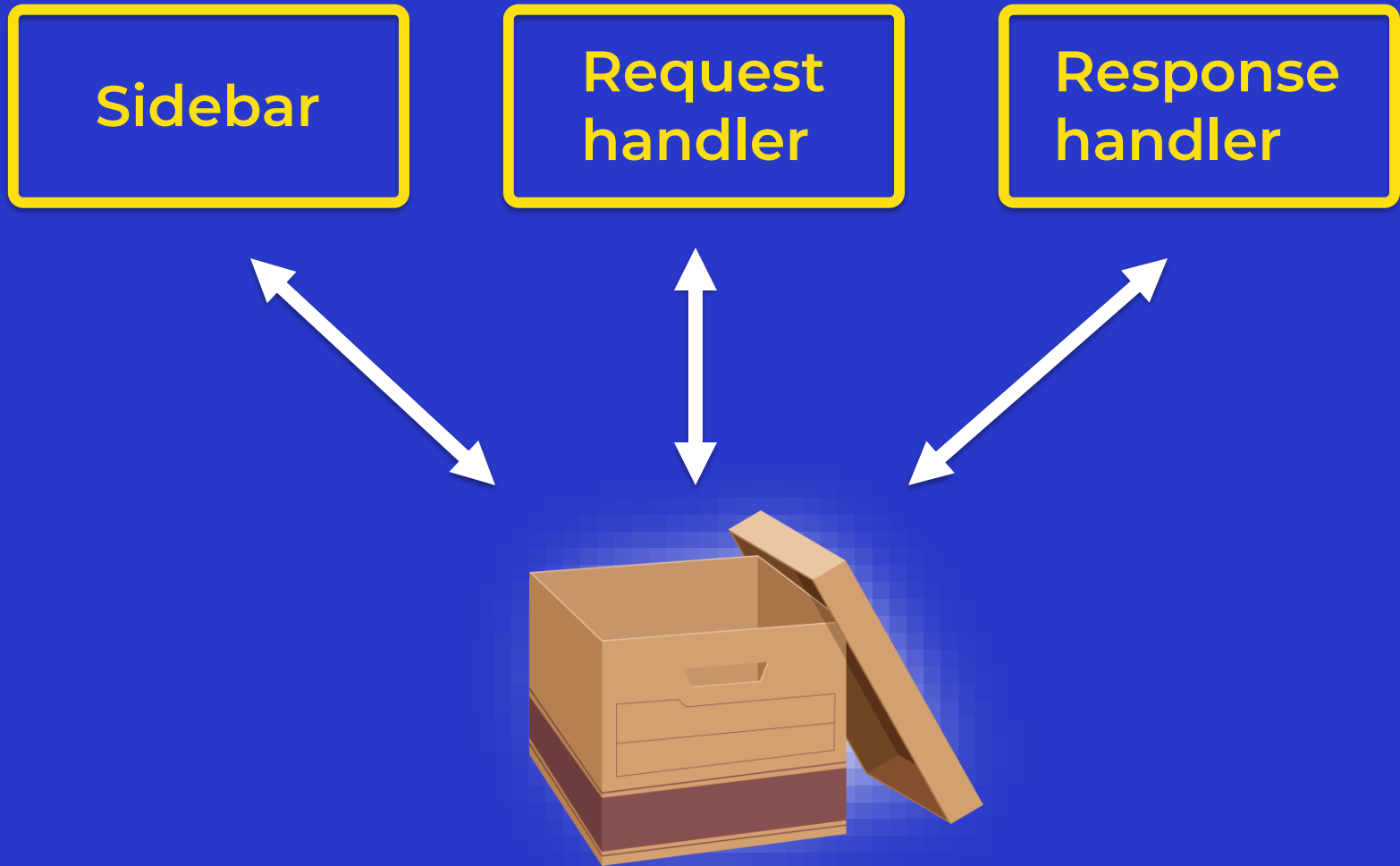
# Svelte's Context

Il **Context** di Svelte permette di condividere dati tra componenti senza doverli passare manualmente come props.

Utilizzato nel progetto per gestire la comunicazione tra:

- **Sidebar** (gestione delle richieste salvate)
- **RequestHandler** (modifica ed invio della richiesta)
- **ResponseHandler** (visualizzazione della risposta)

## Context – how it works



## Context – how it works

In generale, utilizzando un context reattivo, ogni modifica allo stato globale viene rilevata automaticamente dai componenti che lo utilizzano, come se tutti guardassero nella stessa scatola condivisa

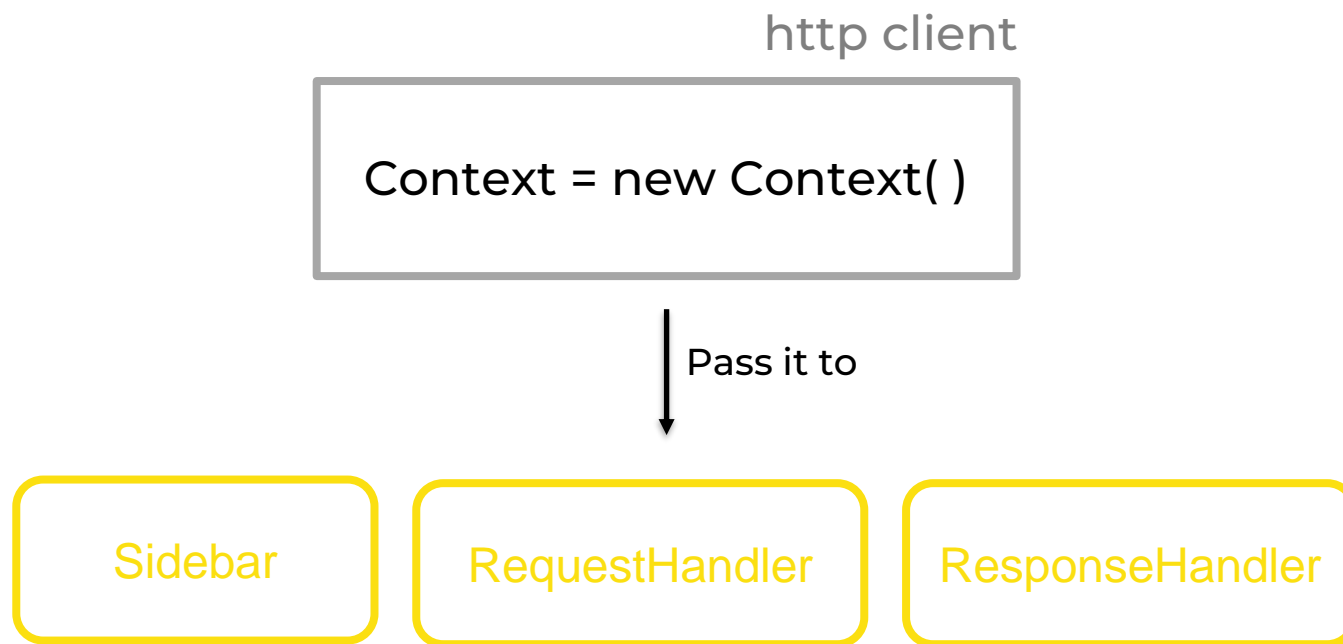
Ma...

Come abbiamo gestito  
l'indipendenza dello  
stato di ogni client?





## Architecture logic



Ognuno dei componenti a cui passiamo il context come prop, accede agli store **indipendenti** attraverso getters e setters.

# Demo Presentation

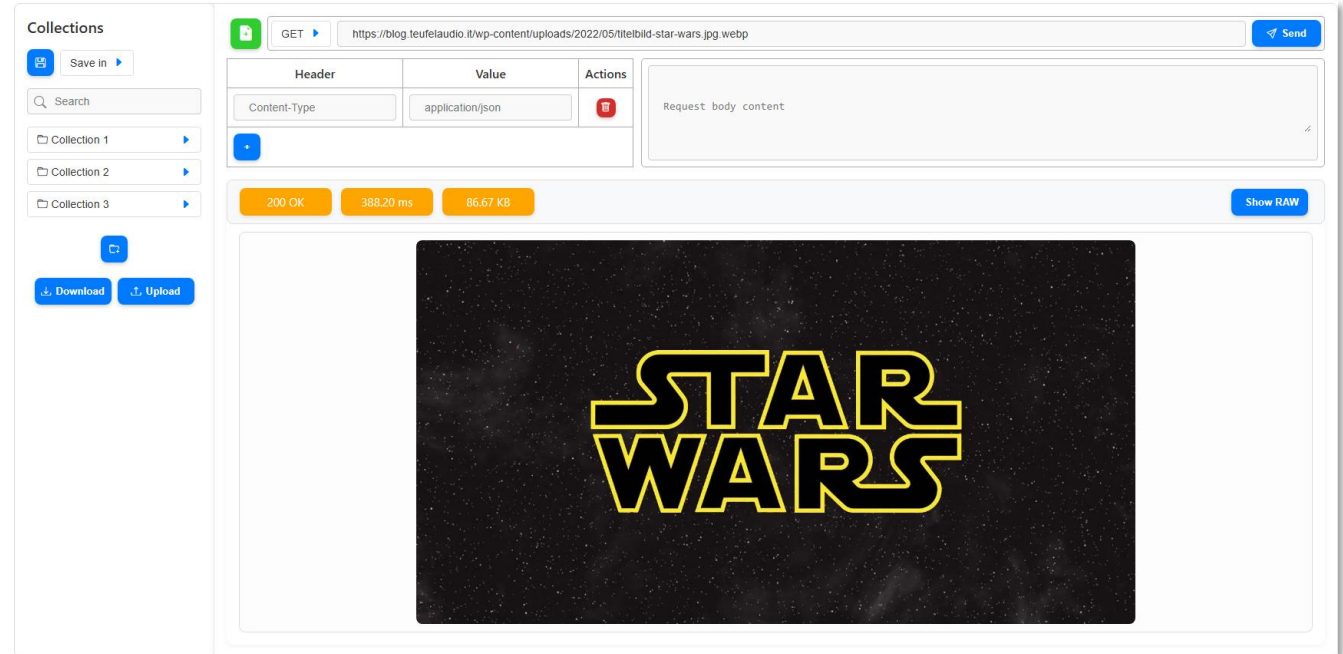
[Clicca qui](#)



## Considerations and conclusions

# Considerations and conclusions

Il progetto ha dimostrato come, grazie a Svelte e a un'architettura ben strutturata, sia possibile costruire un'interfaccia moderna, reattiva e modulare, mantenendo il codice pulito, scalabile e facile da mantenere.



**Grazie!**