

SUPSI

Postman with React

Studente/i

Dominik Panzarella
Alessandro Spagnuolo

Corso di laurea

Ingegneria Informatica

Modulo / Codice Progetto

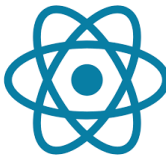
Web Applications 2

Anno

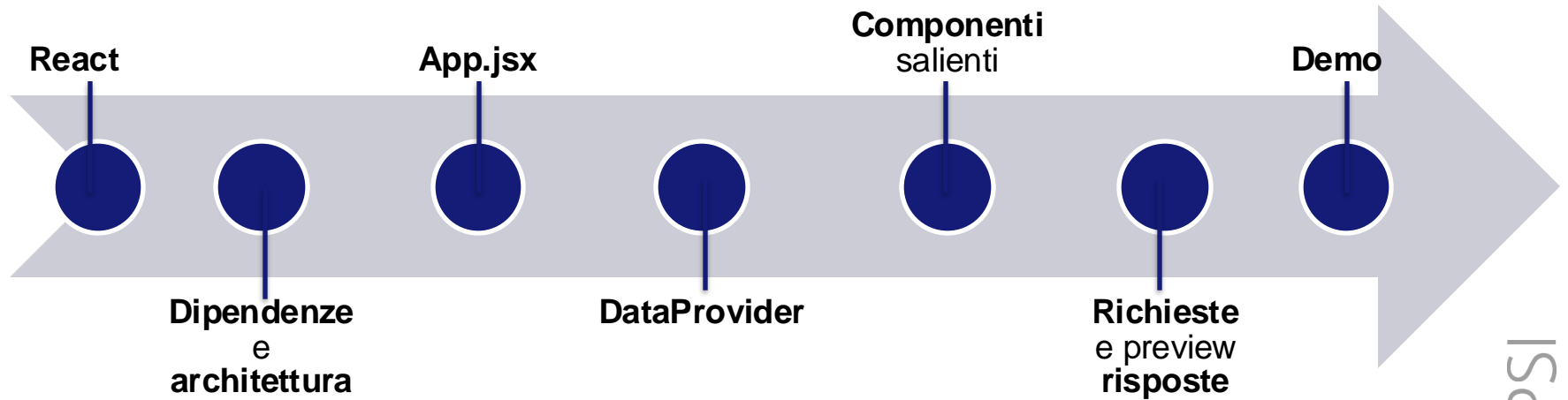
2024-2025

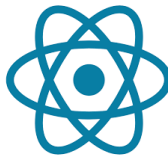
Data

08.04.2025

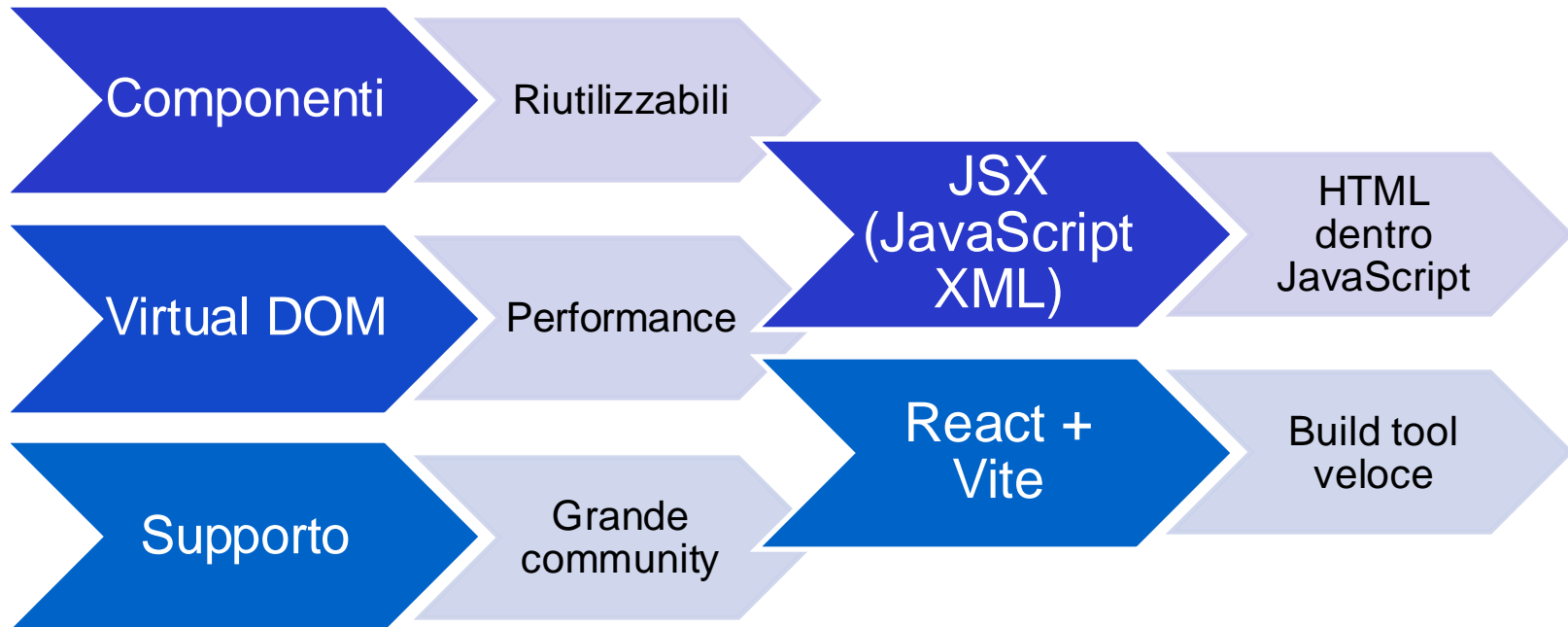


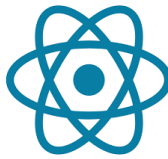
Indice



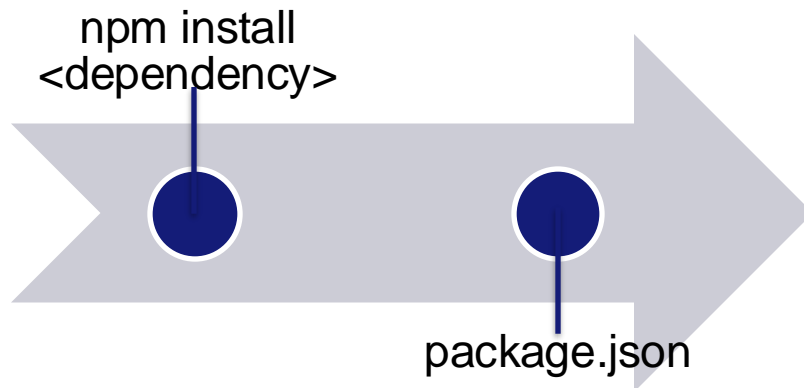


React

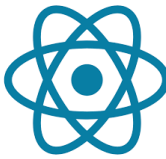




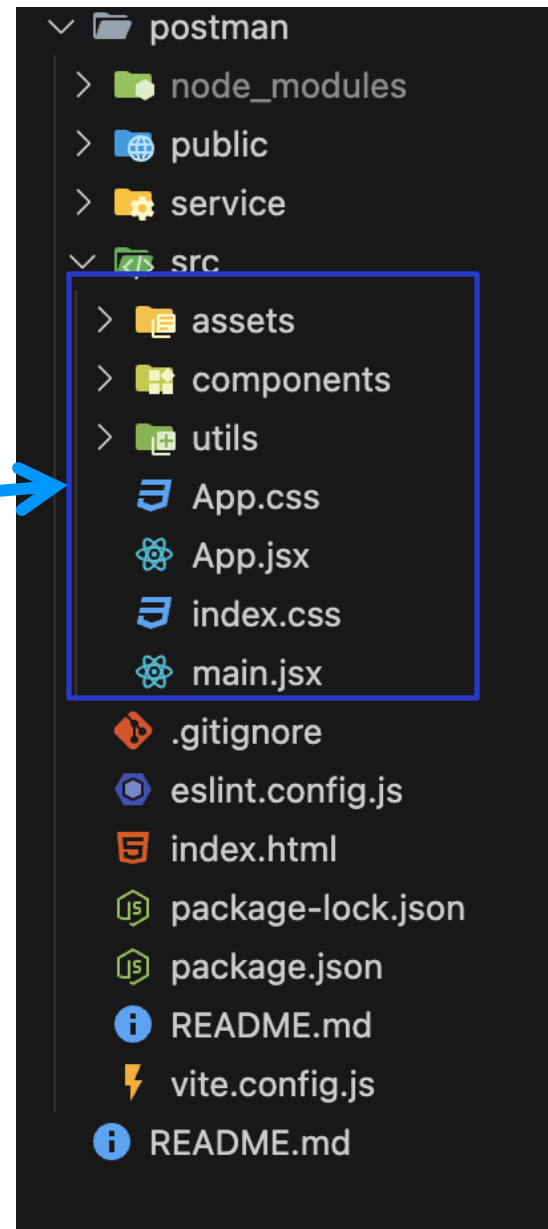
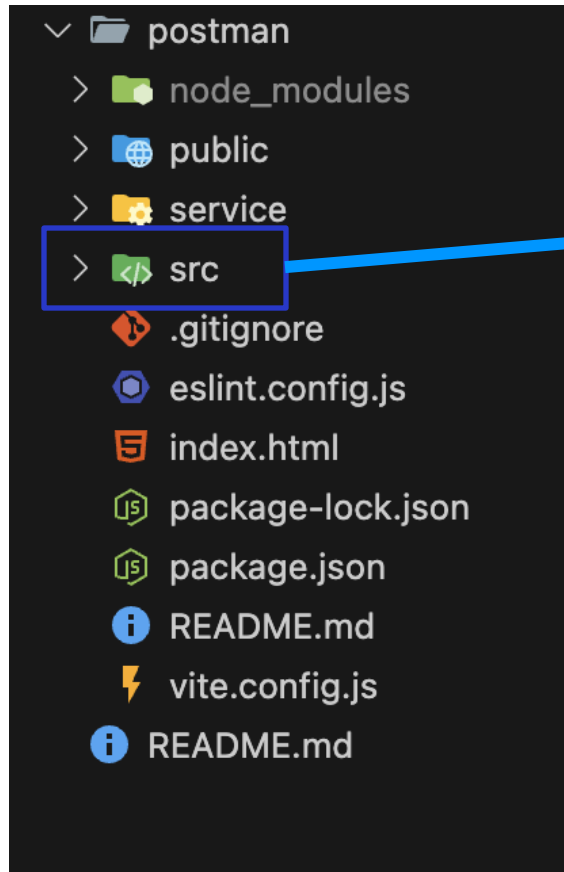
Dipendenze

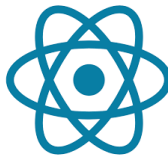


```
1  "dependencies": {  
2    "@emotion/react": "^11.14.0",  
3    "@emotion/styled": "^11.14.0",  
4    "@mui/icons-material": "^6.4.6",  
5    "@mui/material": "^6.4.7",  
6    "@mui/styles": "^6.4.6",  
7    "axios": "^1.8.2",  
8    "react": "^19.0.0",  
9    "react-dom": "^19.0.0",  
10   "react-icons": "^5.5.0",  
11   "react-router-dom": "^7.2.0"  
12 },
```

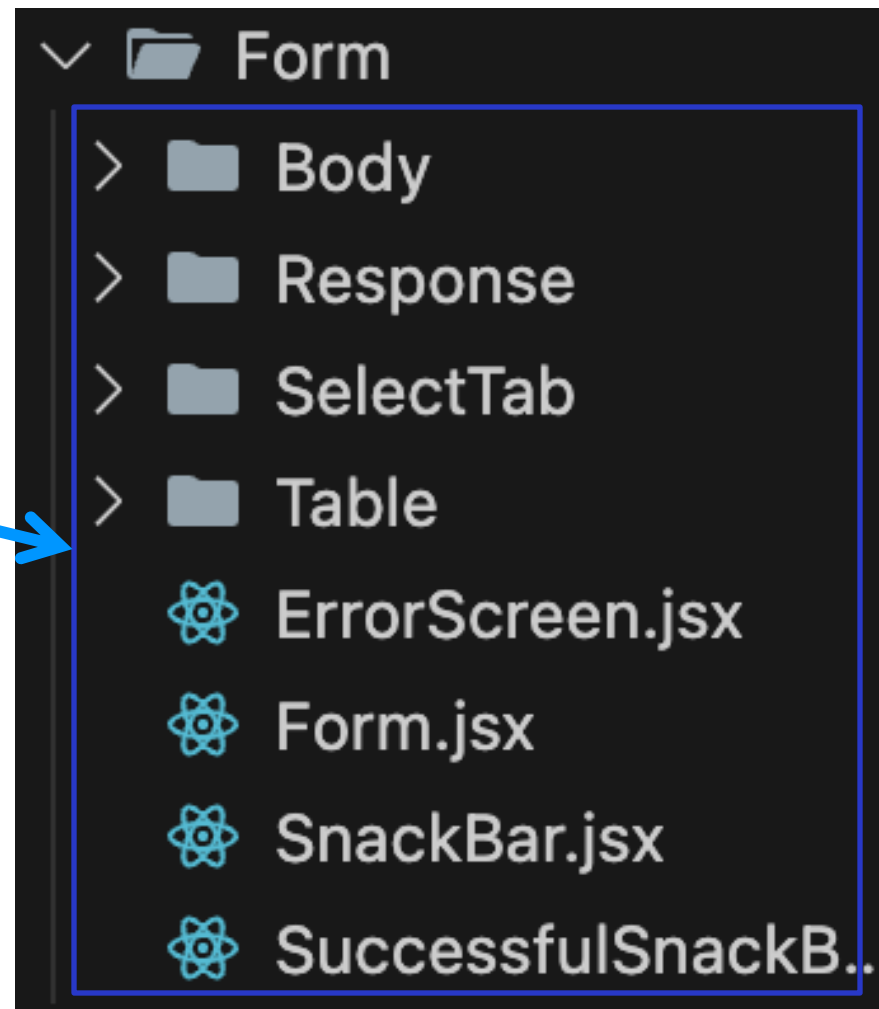
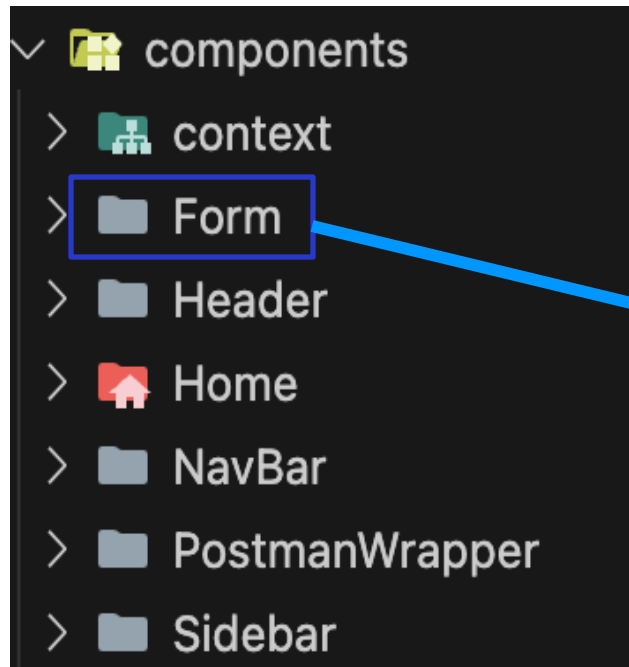


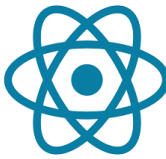
Architettura





Architettura





App.jsx:

Utilizzo di PostmanWrapper

Configure PostmanWrapper

API URL

<https://supsi-ticket.cloudns.org/supsi-http-client/bff>

- ☒ Enable Search bar
- ☒ Enable Sidebar with Collections

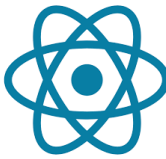
Action on Response Click

Show alert with the status code



```
1 <Box sx={{ mt: 4, width: '100%' }}>
2   <PostmanWrapper
3     url={config.url}
4     search={config.search}
5     collections={config.collections}
6     onResponseMessageClick={responseHandlers[config.responseHandler]}
7   />
8 </Box>
```

props

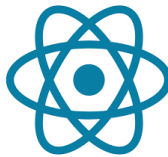


Componente PostmanWrapper

```
1 import DataProvider from '../context/DataProvider';
2 import Home from '../Home/Home';
3
4 function PostmanWrapper({url, search, collections, onResponseMessageClick}) {
5   return (
6     <DataProvider>
7       <Home url={url}
8         search={search}
9         collections={collections}
10        onResponseMessageClick={onResponseMessageClick}>
11     </Home>
12   </DataProvider>
13 );
14 }
15 export default PostmanWrapper;
```

Funzioni dei
componenti
da importare

Funzione del
componente
da esportare



DataProvider



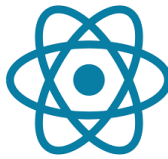
```
1 import React, { useState, createContext } from 'react';
2 export const DataContext = createContext(null);
```



```
1 const DataProvider = ({ children }) => {
2   const [formData, setFormData] = useState({ url: '', type: 'POST' });
3   const [jsonText, setJsonText] = useState('');
4   const [paramData, setParamData] = useState(...);
5   const [headerData, setHeaderData] = useState(...);
6   const [collectionsList, setCollectionsList] = useState([]);
7   const [currentRequest, setCurrentRequest] = useState([]);
8   const [currentCollection, setCurrentCollection] = useState([]);
9   const [dirty, setDirty] = useState(0);
```

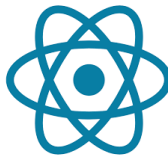


```
1   return (
2     <DataContext.Provider
3       value={{
4         formData, setFormData,
5         jsonText, setJsonText,
6         paramData, setParamData,
7         headerData, setHeaderData,
8         collectionsList, setCollectionsList,
9         currentRequest, setCurrentRequest,
10        currentCollection, setCurrentCollection,
11        dirty, setDirty
12      }}
13     >{children}
14   </DataContext.Provider>
15 )
16 }
17 export default DataProvider;
```

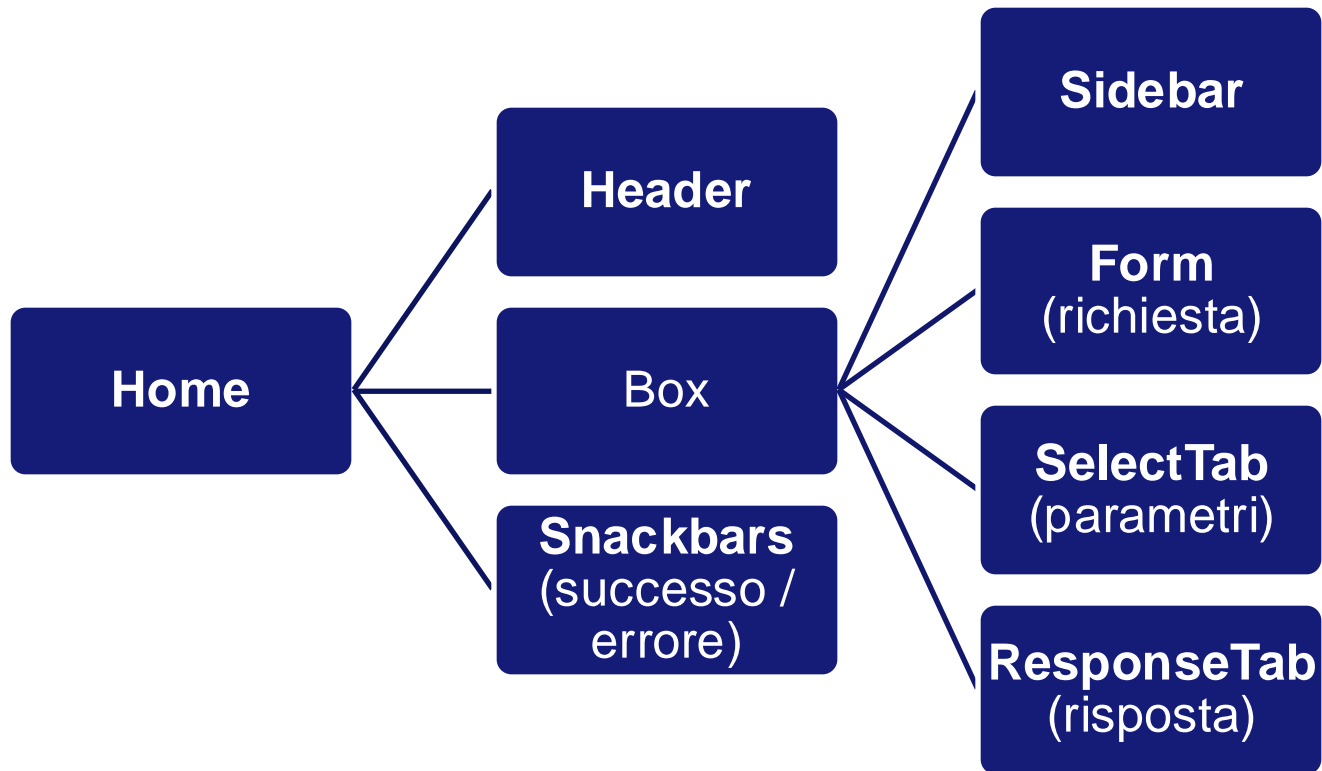


DataProvider - Example

```
1 function PostmanWrapper({url, search, collections, onResponseMessageClick}) {  
2   return (  
3     <DataProvider>  
4       <Home  
5         url={url} search={search}  
6         collections={collections}  
7         onResponseMessageClick={onResponseMessageClick}>  
8       </Home>  
9     </DataProvider>  
10  );  
11 }
```



Componente Home



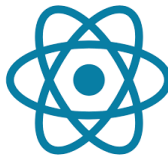
Componente Home - sezione con la sidebar

```
1 {collections && (  
2   <Box sx={{...}}>  
3     <Box sx={{ width: '30%', minWidth: '250px', flexShrink: 0 }}>  
4       <Sidebar url={url} search={search}/>  
5     </Box>  
6     <Box sx={{...}}>  
7       <Form onSendClick={onSendClick} showSnackbar={showSnackbar}  
8         setError={setError} setSuccess={setSuccess}/>  
9       <SelectTab />  
10      {errorResponse ? <ErrorScreen /> :  
11        <ResponseTab apiResponse={apiResponse}  
12          onResponseMessageClick={onResponseMessageClick}/>  
13      </Box>  
14    </Box>  
15  )}
```

Check se mostrare la sidebar

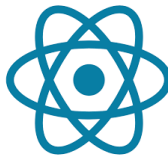
Manda avanti se mostrare la search bar o no

Manda avanti la funzione da chiamare al click sulla risposta



Componente Home - sezione senza la sidebar

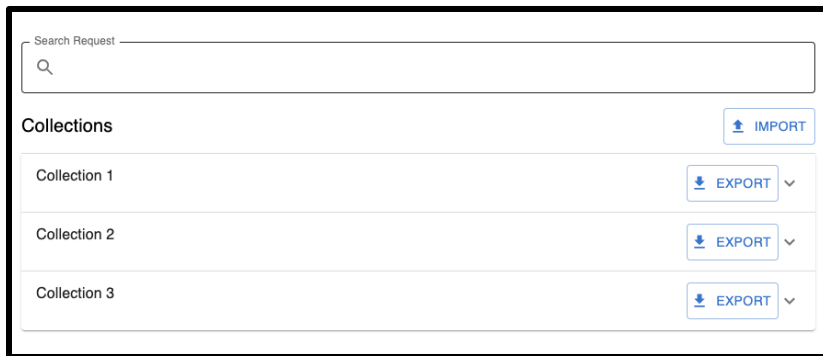
```
1  {!collections && (  
2    <Box sx={{...}}>  
3      <Form onSendClick={onSendClick} showSnackbar={showSnackbar}  
4        setError={setError} setSuccess={setSuccess}/>  
5      <SelectTab />  
6      {errorResponse ? <ErrorScreen /> :  
7        <ResponseTab apiResponse={apiResponse}  
8          onResponseMessageClick={onResponseMessageClick}/>}  
9    </Box>  
10 )}
```



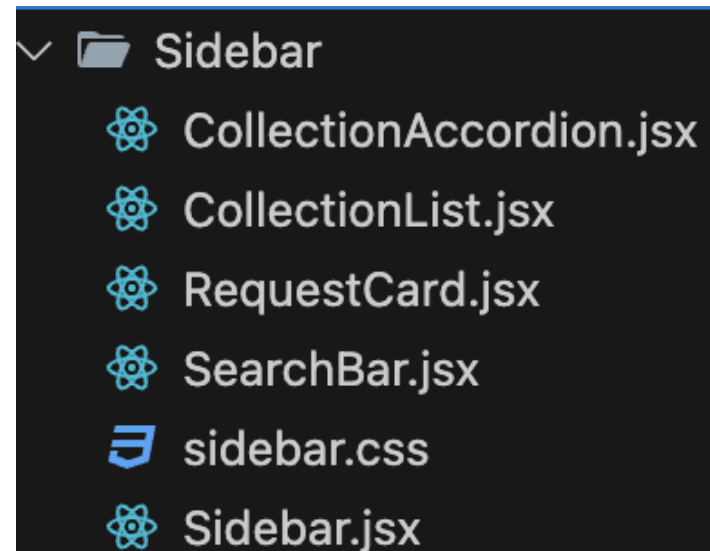
Componente Sidebar



PostmanWrapper



Sidebar



Componente Sidebar



PostmanWrapper

Search Request

Q

Collections

IMPORT

Collection 1	EXPORT
Collection 2	EXPORT
Collection 3	EXPORT



```

1  return (
2    <div className="http-client">
3      {collections && (
4        <div className="collections-sidebar">
5
6          {search && (
7            <SearchBar
8              onSearch={handleSearchChange}
9              searchTerm={searchTerm}></SearchBar>
10          )}
11
12          <input
13            type="file"
14            ref={fileInputRef}
15            style={{ display: 'none' }}
16            accept=".json"
17            onChange={processImportedFile}
18          />
19          <CollectionList
20            collectionsList={collectionsList}
21            expandedCollection={expandedCollection}
22            handleAccordionChange={handleAccordionChange}
23            handleExport={handleExport}
24            handleImport={handleImport}
25            filteredRequests={filterRequests()}
26            handleCardClick={handleCardClick}
27            classes={classes}
28          />
29        </div>
30      )}
31    </div>
32  );
  
```

Searchbar

Componente Sidebar



PostmanWrapper

Search Request

Collections

IMPORT

Collection 1

EXPORT

Collection 2

EXPORT

PUT | Bag-End

PUT | Rivendell

DELETE | Last Desert

PUT | unknown

GET | unknown

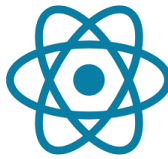
Collection 3

EXPORT

```

1  return (
2    <div className="http-client">
3      {collections && (
4        <div className="collections-sidebar">
5
6          {search && (
7            <SearchBar
8              onSearch={handleSearchChange}
9              searchTerm={searchTerm}></SearchBar>
10          )}
11
12          <input
13            type="file"
14            ref={fileInputRef}
15            style={{ display: 'none' }}
16            accept=".json"
17            onChange={processImportedFile}
18          />
19
20          <CollectionList
21            collectionsList={collectionsList}
22            expandedCollection={expandedCollection}
23            handleAccordionChange={handleAccordionChange}
24            handleExport={handleExport}
25            handleImport={handleImport}
26            filteredRequests={filterRequests()}
27            handleCardClick={handleCardClick}
28            classes={classes}
29          />
30        </div>
31      )}
32    </div>
33  );

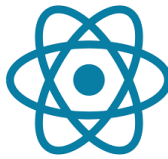
```

Componente Sidebar

CollectionAccordation ←

```
1  const CollectionList = ({
2    collectionsList,
3    expandedCollection,
4    handleAccordionChange,
5    handleExport,
6    handleImport,
7    filteredRequests,
8    handleCardClick,
9    classes
10 }) => {
11   return (
12     <>
13       <div style={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', marginBottom: '10px' }}>
14         <Typography variant="h6">Collections</Typography>
15         <Button
16           className={classes.button}
17           variant="outlined"
18           startIcon={<FileUploadIcon />}
19           onClick={() => handleImport()}
20         >
21           Import
22         </Button>
23       </div>
24
25       {collectionsList.map((collection) => {
26         <CollectionAccordion
27           key={collection.id}
28           collection={collection}
29           expanded={expandedCollection === collection.id}
30           onChange={handleAccordionChange(collection.id)}
31           onExport={() => handleExport(collection)}
32           requests={filteredRequests[collection.id] || []}
33           onRequestClick={handleCardClick}
34           classes={classes}
35         />
36       })}
37     </>
38   );
39 }
```



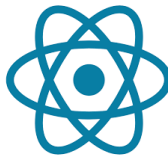
Componente Sidebar

RequestCard

```

1
2 const CollectionAccordion = ({
3   collection,
4   expanded,
5   onChange,
6   onExport,
7   requests,
8   onRequestClick,
9   classes
10 }) => {
11   return (
12     <Accordion
13       expanded={expanded}
14       onChange={onChange}
15       sx={{
16         backgroundColor: expanded ? "#f5f5f5" : "white",
17         transition: "background-color 0.3s ease"
18       }}
19     >
20       <AccordionSummary
21         component="div"
22         expandIcon={<ExpandMore />}
23         aria-controls={`panel-${collection.id}-content`}
24         id={`panel-${collection.id}-header`}
25       >
26         <div style={{ display: 'flex', justifyContent: 'space-between', width: '100%' }}>
27           <Typography variant="body1">{collection.name}</Typography>
28           <Button
29             className={classes.button}
30             variant="outlined"
31             startIcon={<FileDownloadIcon />}
32             onClick={(e) => {
33               e.stopPropagation();
34               onExport();
35             }}
36           >
37             Export
38           </Button>
39         </div>
40       </AccordionSummary>
41       <AccordionDetails>
42         <Grid2 container direction="column" spacing={2}>
43           {requests.map((request) => (
44             <RequestCard request={request} onClick={() => onRequestClick(request)} />
45           ))}
46         </Grid2>
47       </AccordionDetails>
48     </Accordion>
49   );
50 };
51
52

```



Componente Sidebar

RequestCard

```
1  return (  
2    <Card sx={{ marginBottom: '10px', cursor: 'pointer' }} onClick={onClick}>  
3      <CardActionArea>  
4        <CardContent>  
5          <Typography gutterBottom variant="h6" component="div" style={{ textWrapMode: 'nowrap' }}>  
6            <strong style={{ color: getColor(request.method).select }}>{request.method}</strong> | {trimName(request.name)}  
7          </Typography>  
8        </CardContent>  
9      </CardActionArea>  
10    </Card>  
11  );
```

Export

Preparazione dei
dati

Conversione in
JSON e creazione
oggetto

Far partire in
automatico il
download

```
1  const handleExport = async (collection) => {
2    try {
3      const requests = await fetchRequests(collection.id);
4      const exportData = {
5        id: collection.id,
6        name: collection.name,
7        requests: requests
8      };
9
10     const jsonString = JSON.stringify(exportData, null, 2);
11     const blob = new Blob([jsonString], { type: "application/json" });
12     const url = URL.createObjectURL(blob);
13
14     const a = document.createElement("a");
15     a.href = url;
16     a.download = `${collection.name}.json`;
17     document.body.appendChild(a);
18     a.click();
19
20     document.body.removeChild(a);
21     URL.revokeObjectURL(url);
22   } catch (error) {
23     console.error("Error during export:", error);
24   }
25 }
```

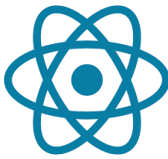
Import

Lettura dal file e
creazione nuova
collection

Aggiornare le
richieste con l'id
collections valido

Aggiornare
lo stato delle
collections

```
1  const handleImport = () => {
2    if (fileInputRef.current) {
3      fileInputRef.current.click();
4    }
5  };
6
7  const processImportedFile = async (event) => {
8    const file = event.target.files[0];
9    if (!file) return;
10   try {
11     const content = await readFileAsJson(file);
12     const nextId = getNextCollectionId();
13     const newCollection = {
14       id: nextId,
15       name: content.name
16     };
17
18     const updatedRequests = content.requests.map(request => ({
19       ...request,
20       collectionId: nextId
21     }));
22     setCollectionsList(prevCollections => [...prevCollections, newCollection]);
23     setRequestsByCollection(prev => ({
24       ...prev,
25       [nextId]: updatedRequests
26     }));
27
28     if (fileInputRef.current) {
29       fileInputRef.current.value = '';
30     }
31   } catch (error) {
32     console.error('Error importing collection:', error);
33     alert('Failed to import collection. Check console for details.');
```



Componente Form



Request's name

POST

SEND >

Headers

Body

Headers

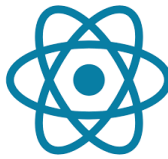
	KEY	VALUE	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	

+

Response




Enter the URL and click **SEND** to get a response.



Componente Form

Richiesta esiste in
una collection*

 **PostmanWrapper**

Search Request

Collections

Collection 1

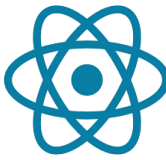
PUT | Ravenhill

GET | TestJSON

Headers Body

Headers

	KEY	VALUE	
<input type="checkbox"/>	Content-Length		<input type="button" value="X"/>
<input type="checkbox"/>	Content-Type	application/json	<input type="button" value="X"/>



Componente Form

Richiesta NON esiste
in una collection

Search Request

Collections

Collection 1	EXPORT
Collection 2	EXPORT
Collection 3	EXPORT

[IMPORT](#)

Request's name
rerererere

PUT

SEND

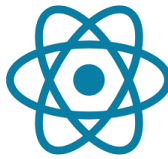
Headers **Body**

Headers

KEY	VALUE

[+](#)

Response



Componente Form

Richiesta
»pendente« in una
collection

Search Request

Request's name
NotAdded [+ CREATE](#)

Collections

Collection 1 [EXPORT](#)

- [PUT](#) | Ravenhill
- [GET](#) | TestJSON
- [GET](#) | known
- [GET](#) | TestPDF
- [GET](#) | TestImage
- [GET](#) | TestHTML
- [POST](#) | TestX
- [POST](#) | unknown

[IMPORT](#) [PUT](#) [SEND](#)


Headers Body

Headers

KEY	VALUE

[+](#)

Response



Enter the URL and click **SEND** to get a response.

Componente Form

PostmanWrapper

Search Request
Q

Collections

- Collection 1 **EXPORT**
- Collection 2 **EXPORT**
- Collection 3 **EXPORT**

Collection 1 **EXPORT**

- PUT** | Ravenhill
- GET** | TestJSON
- GET** | known
- GET** | TestPDF
- GET** | TestImage
- GET** | TestHTML
- POST** | TestX
- POST** | unknown


Request's name
Ravenhill

PUT <https://supsi-bucket.cloudns.org/supsi-http-client/test-endpoints/lotr/random-character> **SEND**

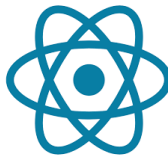
Headers

KEY	VALUE
Content-Length	
Content-Type	application/json

Response

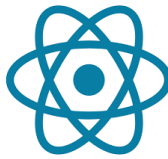

Enter the URL and click SEND to get a response.

Collection importata



```
1      <Box sx={{ paddingBottom: 0 }}>
2        {currentCollection && Number(currentCollection) > 0 && Number(currentCollection) <= 3 && currentCollection !== undefined && (
3          <>
4            {currentRequest.id && (
5              <Button
6                className={classes.button}
7                variant="outlined"
8                startIcon={SaveAltIcon} />
9                onClick={() => onSaveClick()}
10              >
11                Save
12              </Button>
13            )}
14            {!currentRequest.id && (
15              <Button
16                className={classes.button}
17                variant="outlined"
18                startIcon={AddIcon} />
19                onClick={() => onSaveClick()}
20              >
21                Create
22              </Button>
23            )}
24            {currentRequest.id && (
25              <>
26                <Button
27                  className={classes.button}
28                  variant="outlined"
29                  color="error"
30                  startIcon={DeleteIcon} />
31                  onClick={() => onDeleteClick()}
32                >
33                  Delete
34                </Button>
35              </>
36            )}
37          </>
38        )
39      </Box>
```

Condizione
Di
Renderizzazione




Gestione delle richieste

Incapsulamento
Dati

Proxy SUPSI

```
1 export const getData = async (formData, jsonText, paramData, headerData) => {
2   const apiType = formData.type;
3   const apiUrl = formData.url;
4   const apiHeaders = getHeadersAndParams(headerData);
5   const apiParams = getHeadersAndParams(paramData);
6   const data = apiType === 'GET' ? '' : jsonText;
7
8   const transformedHeaders = Object.entries(apiHeaders).reduce((acc, [key, value]) => {
9     acc[key] = Array.isArray(value) ? value : [String(value)];
10    return acc;
11  }, {});
12
13  const supsiData = {
14    method: apiType,
15    uri: apiUrl,
16    headers: transformedHeaders,
17    body: data
18  };
19
20  try {
21    const response = await axios({
22      method: 'POST',
23      url: 'https://supsi-ticket.cloudns.org/supsi-http-client/proxy/execute',
24      data: supsiData,
25      headers: {'Content-Type': ['application/json'], 'Accept': ['application/json']},
26    });
27
28    return response;
29  } catch (error) {
30    console.log('Error while getting the response: ', error);
31    return {
32      error: true,
33      message: error.message || 'An error occurred'
34    };
35  }
36 }
37
```

Componente Form





 Request's name

POST

Headers

Body

Headers

	KEY	VALUE	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	


Response

SelectTab component



Enter the URL and click **SEND** to get a response.





Componente Form

 Request's name

POST **SEND** ➤


Headers **Body**

Headers

	KEY	VALUE	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	

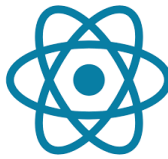
+

Response



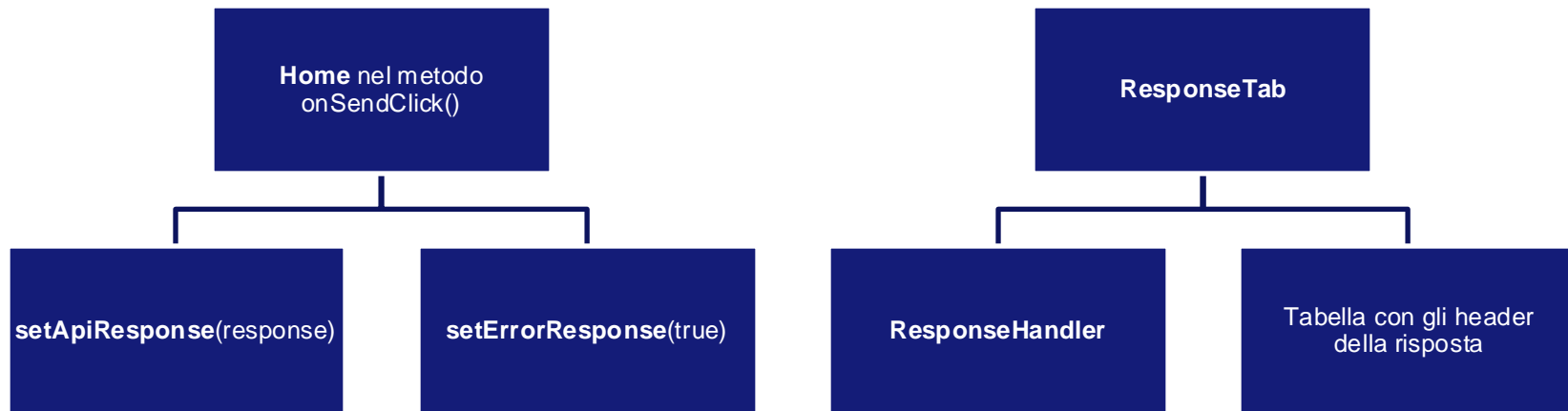
Enter the URL and click **SEND** to get a response.

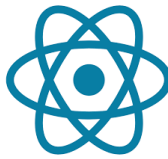
ResponseHandler
component



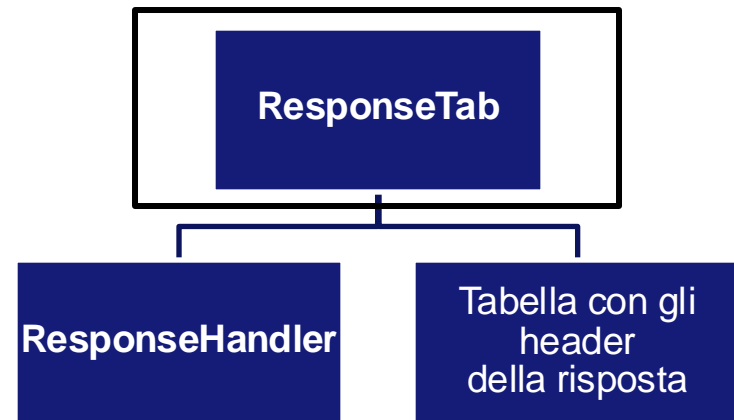
Gestione delle risposte

```
{errorResponse ? <ErrorScreen /> :  
  <ResponseTab apiResponse={apiResponse}  
    onResponseMessageClick={onResponseMessageClick}/>}
```





Gestione delle risposte



BODY

HEADERS

200 OK

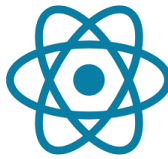
98 MS

23.47 KB

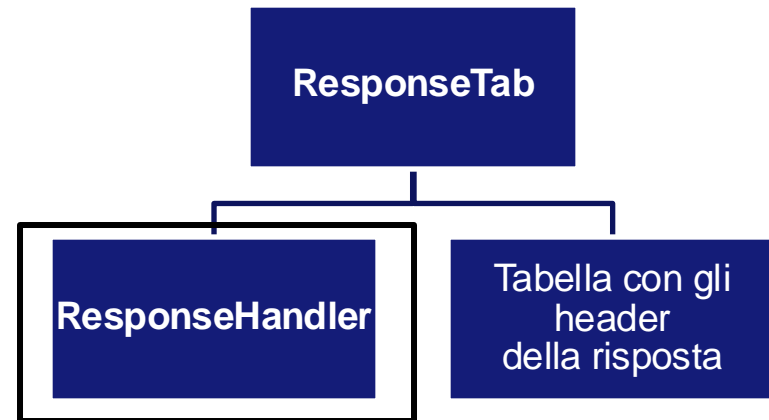


```

1 <Box role="tabpanel" hidden={value !== 0}
2   id={`simple-tabpanel-${0}`}
3   aria-labelledby={`simple-tab-${0}`}>
4   <ResponseHandler apiResponse={apiResponse} onResponseMessageClick={onResponseMessageClick}></ResponseHandler>
5 </Box>
6 <Box role="tabpanel" hidden={value !== 1}
7   id={`simple-tabpanel-${1}`}
8   aria-labelledby={`simple-tab-${1}`}>
9   <CreateTable text="" data={convertArray(apiResponse.data.headers)} setData={() => {}} modifiable={false} />
10 </Box>
  
```

Gestione delle risposte



```
1 import HTMLHandler from "../HTML/HTMLHandler";
2 import IMAGEHandler from "../IMAGE/IMAGEHandler";
3 import JSONHandler from "../JSON/JSONHandler";
4 import PDFHandler from "../PDF/PDFHandler";
5
6 const ResponseHandler = ({apiResponse, onResponseMessageClick}) => {
7   const responseHeader = apiResponse.data.headers || '';
8   const contentTypeArray = responseHeader['Content-Type'] || '';
9   const contentTypeHeader = contentTypeArray[0];
10  const body = apiResponse.data.body || '';
11
12  if(contentTypeHeader.includes('json'))
13  {
14    return (<div onClick={()=>onResponseMessageClick(apiResponse)}><JSONHandler apiResponse={apiResponse}></JSONHandler></div>);
15  }else if(contentTypeHeader.includes('text/html') || (typeof data === 'string' && data.trim().startsWith('<'))){
16    return (<div onClick={()=>onResponseMessageClick(apiResponse)}><HTMLHandler apiResponse={apiResponse}></HTMLHandler></div>);
17  }else if (contentTypeHeader.includes('image/')) {
18    return (<div onClick={()=>onResponseMessageClick(apiResponse)}><IMAGEHandler apiResponse={apiResponse}></div>);
19  }else if (contentTypeHeader.includes('application/pdf')) {
20    return (<div onClick={()=>onResponseMessageClick(apiResponse)}><PDFHandler apiResponse={apiResponse}></div>);
21  }
22 };
23 export default ResponseHandler;
```

Import e uso dei relativi handler

Click sulla risposta

Preview delle risposte - pdf

```
1 const PDFHandler = ({ apiResponse }) => {
2   const classes = useStyles();
3   const [pdfUrl, setPdfUrl] = useState('');
4
5   useEffect(() => {
6     if (apiResponse) {
7       convertPDF();
8     }
9     return () => {
10      if (pdfUrl) {
11        URL.revokeObjectURL(pdfUrl);
12      }
13    };
14  }, [apiResponse]);
```

Request's name
TestPDF

SAVE DELETE

GET SEND

Headers Body

Headers

KEY	VALUE
KEY	VALUE

All'arrivo di una
nuova risposta

Cancella l'oggetto
creato per il vecchio
pdf

Preview delle risposte - pdf

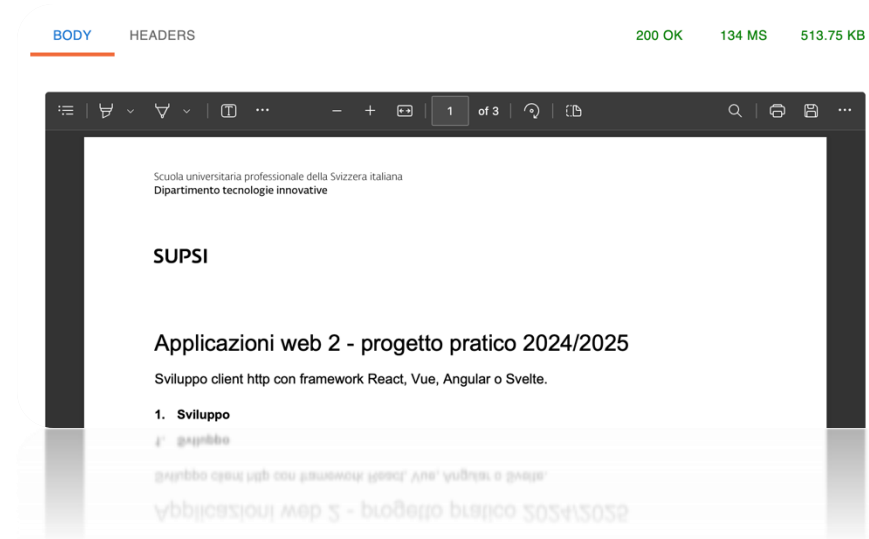
```
1  const convertPDF = () => {
2    const headers = apiResponse.data.headers || '';
3    const contentTypeArray = headers['Content-Type'] || '';
4    const contentTypeHeader = contentTypeArray[0];
5    let pdfContent = apiResponse.data.body;
6
7    if (!pdfContent) {
8      console.error('No content in the API response');
9      return;
10   }
11   try {
12     const byteCharacters = atob(pdfContent);
13     var byteArray = new ArrayBuffer(byteCharacters.length);
14     var integerArray = new Uint8Array(byteArray);
15
16     for (let i = 0; i < byteCharacters.length; i++) {
17       integerArray[i] = byteCharacters.charCodeAt(i);
18     }
19
20     const blob = new Blob([integerArray], { type: contentTypeHeader });
21     const blobUrl = URL.createObjectURL(blob);
22     setPdfUrl(blobUrl);
23   } catch (error) {
24     console.error('Error converting image:', error);
25   }
26   };
```

atob() e parsing del
contenuto della risposta

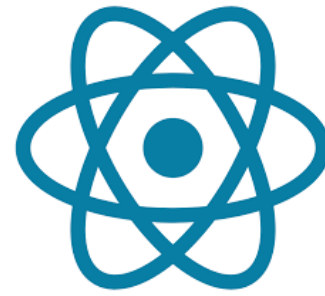
Creazione
dell'oggetto e dell'url
per il pdf

Preview delle risposte - pdf

```
1 return (  
2   <Box className={classes.container}>  
3     {pdfUrl ? (  
4       <iframe  
5         src={pdfUrl}  
6         title="PDF Preview"  
7         className={classes.pdfFrame}  
8       />  
9     ) : (  
10      <p>Loading PDF...</p>  
11    )}  
12  </Box>  
13 );
```



iframe a cui
passare l'url
del pdf da
visualizzare



Demo

