UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA

Dipartimento di Scienze Fisiche, Informatiche e Matematiche

Corso di Laurea in Informatica

Tesi di Laurea

Progettazione e sviluppo di Sophon, applicativo cloud a supporto della ricerca

Relatore:

Prof.ssa
Claudia Canali

Candidato:

Matr. 128570
Stefano Pigozzi

Anno Accademico 2020-2021

Contenuti

CAPITOLO 1

Sinossi

La tesi descrive lo sviluppo di **Sophon**, software che permette a gruppi di ricerca di collaborare in sicurezza da remoto all'interno di ambienti *Jupyter* eseguiti sui server della loro organizzazione.

Figura 1.1: Cattura schermo dell'interfaccia grafica di Sophon.

CAPITOLO 2

Introduzione alla tesi

Nel mondo della ricerca universitaria, è frequente la collaborazione tra più colleghi su uno stesso progetto, condividendo dati e materiale al fine di ottenere risultati più efficientemente.

Negli ultimi decenni, le possibilità per effettuare ricerca collaborativa si sono moltiplicate, grazie principalmente allo sviluppo della rete Internet, che permette comunicazione immediata tra istituzioni di tutto il mondo.

Internet infatti ha portato alla creazione di strumenti informatici per facilitare la comunicazione e collaborazione remota, come email, chat, strumenti di revisione bozze e sistemi di controllo versione.

Parallelamente a questi strumenti, sono stati sviluppati degli ambienti di lavoro a supporto della ricerca, che permettono la creazione di documenti interattivi, detti *notebook*, che combinano analisi di dati ad elementi multimediali come testo formattato, immagini e grafici.

Il più popolare di questi ambienti di lavoro è l'ambiente computazionale *Jupyter*, che all'interno del notebook permette l'utilizzo di molteplici linguaggi di programmazione, sia per l'elaborazione dei dati, sia per la stesura del testo.

Jupyter però ha alcuni problemi: è complesso da installare e mantenere, e non implementa di default molte funzionalità per la collaborazione, la confidenzialità e l'autenticazione, che devono quindi essere implementate da strumenti esterni.

2.1 Obiettivo della tesi

L'obiettivo di questa tesi è quello di descrivere lo sviluppo dell'applicativo *"Sophon"*, realizzato con il fine di semplificare l'utilizzo di *Jupyter* in ambiente universitario.

2.2 Struttura della tesi

La tesi è strutturata nel seguente modo:

1. nel primo capitolo, *Sinossi*, viene descritto molto brevemente il progetto realizzato;
2. nel secondo capitolo, *Introduzione alla tesi*, viene introdotto il contesto della tesi, la tesi stessa e i suoi contenuti;
3. nel terzo capitolo, *Ricerca collaborativa*, viene presentata in dettaglio la situazione attuale della ricerca collaborativa;
4. nel quarto capitolo, *Progettazione di Sophon*, viene descritta la progettazione avvenuta, entrando nei dettagli dei requisiti, della suddivisione in moduli e delle astrazioni create;
5. nel quinto capitolo, *Realizzazione di Sophon*, vengono trattate le specifiche tecniche implementative del progetto;
6. nel sesto capitolo, *Risultati ottenuti*, viene mostrato il risultato finale del processo di sviluppo;
7. nel settimo capitolo, *Il futuro di Sophon*, vengono tratte le conclusioni della tesi.

In aggiunta, nell'appendice, è disponibile una guida tecnica all'*Installazione di Sophon*.

CAPITOLO 3

Ricerca collaborativa

Nelle scienze, sia teoriche, sia sperimentali, si verifica spesso la necessità di dover prendere appunti e condividere appunti sulla ricerca effettuata.

Mentre in passato a tale scopo venivano utilizzati quaderni di carta (detti anche *"blocchi note laboratoriali"* [wiki:eln]), con la nascita dell'informatica si iniziarono ad utilizzare strumenti digitali, più comodi ed efficienti: inizialmente, semplici sistemi di composizione tipografica come *TeX*, poi editor WYSIWYG (What You See Is What You Get) come *Microsoft Word*, arrivando infine negli ultimi anni ai più avanzati e interattivi *notebook computazionali*.

3.1 Sistemi di composizione tipografica

I primi sistemi utilizzati in ambito accademico per la creazione di documenti erano molto semplici: si limitavano a descrivere come dovevano apparire i contenuti sul foglio stampato attraverso istruzioni molto simili a quelle di un linguaggio di programmazione.

Sono esempi di sistemi di composizione tipografica TeX[1], usato ancora oggi in combinazione con il sistema LaTeX[2] per comporre documenti accademici come paper e tesi (inclusa questa), e roff[3], su cui si basa oggi lo strumento groff[4] per comporre le pagine di manuale dei sistemi operativi Unix-like.

Un esempio di documento LaTeX [overleaf:learn30mins] è il seguente:

```
\documentclass{article}

\begin{document}
```

(continues on next page)

[1] https://www.tug.org/begin.html
[2] https://www.latex-project.org//
[3] https://en.wikipedia.org/wiki/Roff_(software)
[4] https://it.wikipedia.org/wiki/Groff_(software)

(continua dalla pagina precedente)

```
   First document.
   This is a simple example, with no extra parameters or packages␣
↪included.
\end{document}
```

3.2 Editor WYSIWYG

Con l'evolversi dei sistemi operativi, in particolare con la diffusione dei sistemi operativi a finestre, sono stati sviluppati software detti "editor di testo WYSIWYG", che permettono di scrivere documenti avendo un'anteprima istantanea del testo inserito.

Essendo molto più intuitivi da usare dei loro predecessori, ne hanno preso rapidamente il posto in tutto il mondo, di fatto limitando l'uso dei *sistemi di composizione tipografica* ad ambiti in cui era necessaria una formattazione avanzata dei documenti.

Alcuni esempi moderni di editor WYSIWYG sono Microsoft Word[5] e LibreOffice Writer[6].

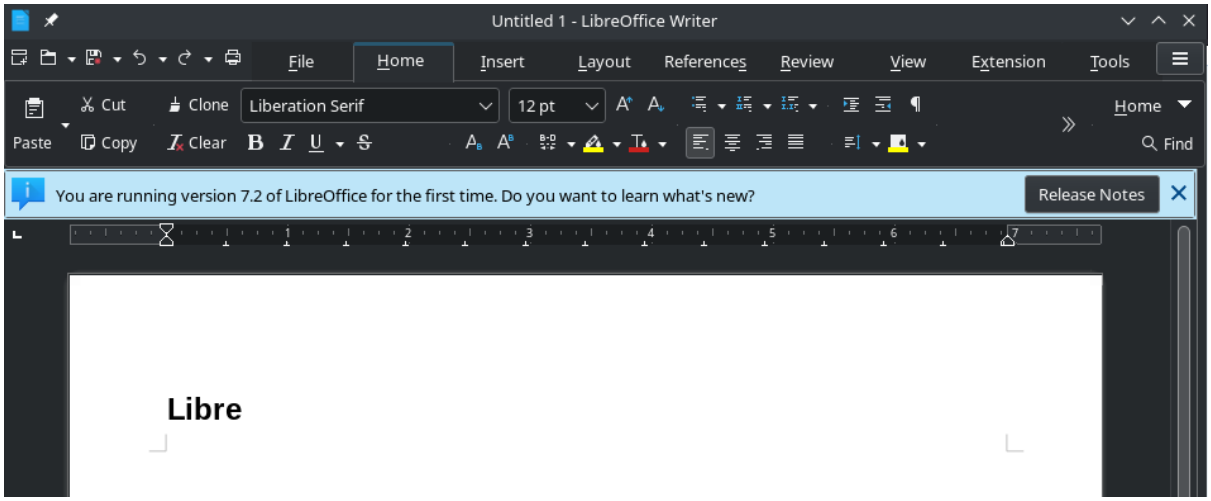


Figura 3.2.1: Modifica di un documento su LibreOffice 7.2.2.2.

3.3 Web-based editor

Il paradigma web "2.0" ha portato miglioramenti significativi agli editor WYSIWYG, rendendoli utilizzabili online come software-as-a-service direttamente da un browser.

Ciò ha semplificato il processo di collaborazione sui documenti: non è più necessario inviare ai collaboratori tutte le revisioni dei documenti, ma è possibile semplicemente condividergli un link, al quale sarà possibile accedere al documento.

[5] https://www.microsoft.com/it-it/microsoft-365/word
[6] https://it.libreoffice.org/scopri/writer/

Da questa funzionalità ne è poi derivata un'altra, che ha rivoluzionato la scrittura di testi: la possibilità di collaborare online con gli altri autori, vedendo le loro modifiche in tempo reale.

Il più importante di questi editor è Google Docs[7], rilasciato nel 2009; la sua popolarità ha portato allo sviluppo di alternative come Office 365[8], una versione web di *Microsoft Word*.

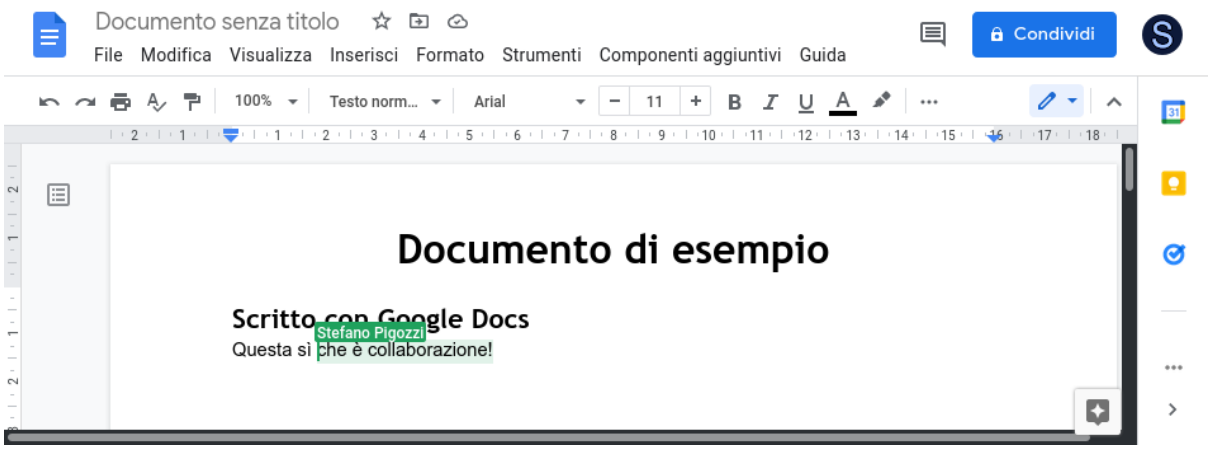


Figura 3.3.1: Un esempio di collaborazione su un documento Google Docs.

3.4 Notebook computazionali

In parallelo ai *web-based editor*, ha preso piede nel mondo della ricerca scientifica una nuova tipologia di documento: il notebook computazionale.

I *notebook computazionali* sono un tipo di documento interattivo che permette contemporaneamente di analizzare dati, elaborarli e documentare elaborazioni effettuate e risultati ottenuti.

Essi sono composti da tante **celle**, ciascuna contenente codice in un determinato linguaggio di programmazione o di marcatura, il quale è eseguito, mostrandone poi i risultati all'utente, sotto forma di testo, equazioni, immagini, grafici, o anche widget interattivi come slider o aree di input testo.

Alcuni esempi di software per la scrittura di notebook computazionali sono Jupyter[9], Wolfram Mathematica[10], MATLAB Live Editor[11]...

[7] https://docs.google.com/
[8] https://www.office.com/
[9] https://jupyter.org/
[10] https://www.wolfram.com/mathematica/
[11] https://it.mathworks.com/products/matlab/live-editor.html

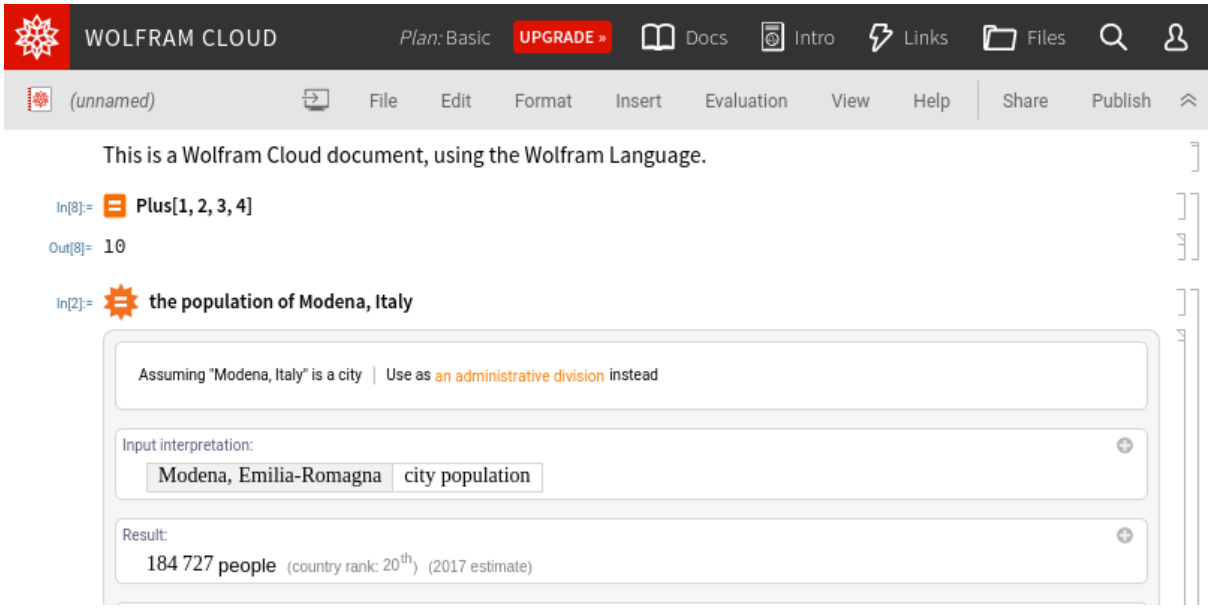


Figura 3.4.1: Un esempio di notebook Mathematica, scritto su Wolfram Cloud.

3.5 Jupyter

Jupyter è un software open-source che permette la scrittura e la visualizzazione di *notebook computazionali*.

Come tutti i notebook computazionali è strutturato in celle, le quali possono contenere testo, dati oppure codice di programmazione con relativo output.

Prende ispirazione dai *web-based editor*, permettendo agli utenti di modificare i notebook direttamente da un browser web, e include rudimentali funzionalità di collaborazione in tempo reale [jupyter:collaboration].

3.5.1 Componenti di Jupyter

Jupyter è composto da 3 componenti: un *kernel*, un *server* e un *client*.

3.5.1.1 Kernel Jupyter

Il kernel è la parte di Jupyter che si occupa di eseguire le celle del notebook, restituendone i risultati al *server*.

Per ogni linguaggio di programmazione che si desidera utilizzare nel notebook è necessario il relativo **kernel**: il kernel predefinito di Jupyter è IPython[12], che permette di utilizzare il linguaggio di programmazione Python[13]; sono però disponibili tanti altri kernel, tra cui uno per Julia[14] e uno per R[15] [jupyter:kernels].

[12] https://ipython.org/
[13] https://www.python.org/
[14] https://julialang.org/
[15] https://www.r-project.org/

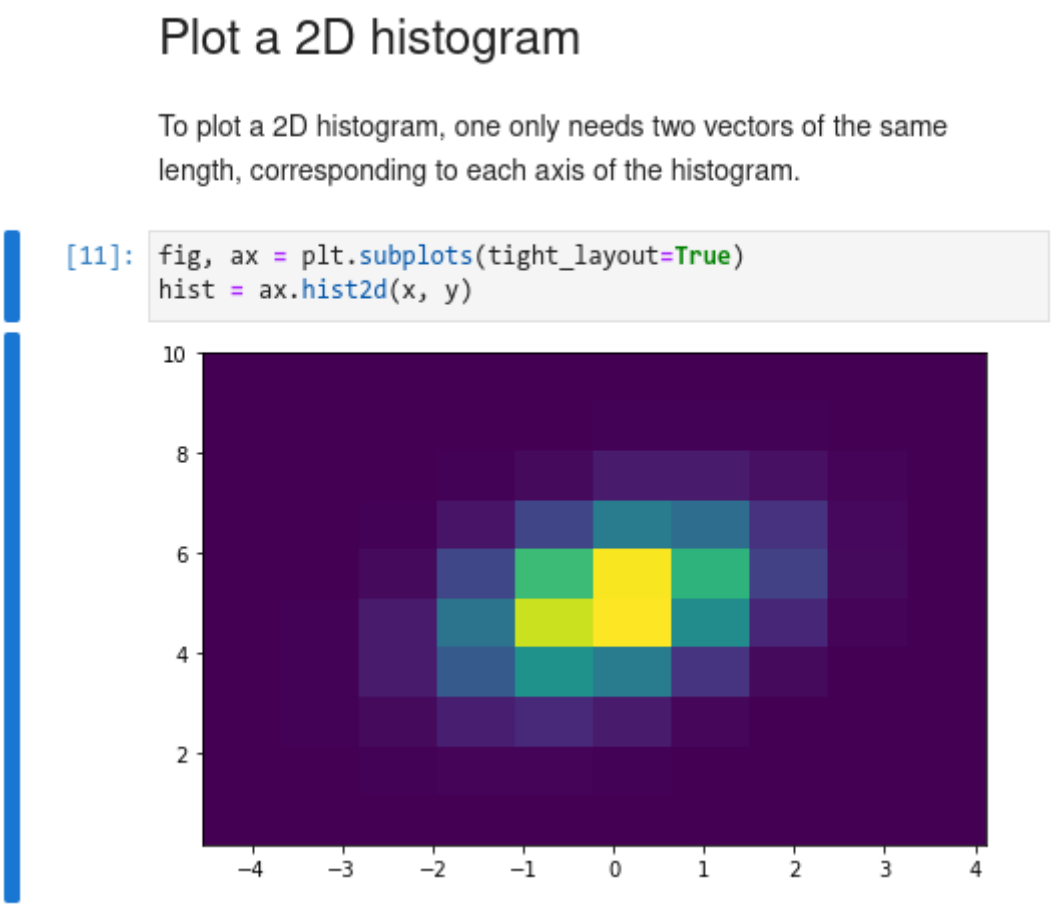


Figura 3.5.1: Un esempio di notebook Jupyter con una cella di testo e una cella di Python che emette un grafico [matplotlib:histograms].

3.5.1.2 Server Jupyter

Il **server** è la parte di Jupyter che gestisce le interazioni del *client* con il notebook, inoltrandole al *kernel* appropriato se necessario.

Il server ufficiale di Jupyter è Jupyter Server[16].

3.5.1.3 Client Jupyter

Il **client** è la parte di Jupyter che mostra in un formato user-friendly il contenuto del notebook e gli permette di interagirvi, comunicando le interazioni al *server*.

Esistono due client ufficiali per Jupyter: il client di vecchia generazione Jupyter Notebook[17] e il client di nuova generazione JupyterLab[18], entrambi web-based.

3.5.2 Hosting di Jupyter

Essendo *server* e *client* separati, è possibile eseguire il server su una macchina e il client su un'altra.

È possibile selezionare la macchina su cui eseguire il server in tre modi diversi, elencati nelle prossime sezioni, ciascuno con alcuni vantaggi e svantaggi.

3.5.2.1 Hosting locale

È possibile installare il server Jupyter **sul proprio computer**.

Così facendo, le celle saranno eseguite con le risorse del proprio computer, e il notebook sarà accessibile solo dal computer che sta eseguendo il server.

È un ottimo modo per lavorare su progetti personali, in quanto offre la massima personalizzazione attraverso un sistema di plugin installabili, e per lavorare offline, in quanto è l'unico modo di usare il server senza connessione ad Internet.

In base al proprio sistema operativo, però, potrebbe risultare difficile da installare, e in base alla propria configurazione di rete, la collaborazione realtime su un progetto potrebbe essere impossibile.

3.5.2.2 Come software-as-a-service

È possibile utilizzare un server Jupyter **gestito da un cloud provider** ed utilizzare le risorse da esso fornite per eseguire le celle.

Un esempio di cloud provider che fornisce questo servizio è Google, con Google Colaboratory[19].

Usare il modello SaaS (Software as a Service) è il modo più semplice per usare Jupyter, in quanto non richiede di effettuare alcuna installazione sul proprio computer, e in genere permette di collaborare online con altri utenti.

[16] https://github.com/jupyter-server/jupyter_server
[17] https://github.com/jupyter/notebook
[18] https://github.com/jupyterlab
[19] https://colab.research.google.com/#

Di contro, però, Jupyter sulle piattaforme SaaS non permette l'installazione di plugin, limitando la personalizzazione, e, se si necessitano più risorse di quelle offerte gratuitamente dai provider, si rischiano canoni mensili elevati.

3.5.2.3 Hosting on-premises

È possibile configurare un **server della propria istituzione** in modo tale che esegua uno o più *server Jupyter* a cui si connetteranno i *client*.

A tale scopo, è disponibile il progetto JupyterHub[20], in grado di gestire migliaia di utenti simultanei [jupyter:ifaq], ciascuno con il proprio notebook.

È performante ed efficace, e in base alla configurazione scelta dall'amministratore, può permettere agli utenti di personalizzare il loro ambiente di lavoro con plugin.

L'interfaccia di gestione utenti e notebook è però molto essenziale, e in aggiunta non supporta nativamente la collaborazione real-time su un singolo notebook, preferendo il modello *"tanti server Jupyter da utente singolo"* [jupyter:hub].

[20] https://jupyter.org/hub

CAPITOLO 4

Progettazione di Sophon

Vista la situazione della *ricerca collaborativa*, si è ritenuto potesse essere utile sviluppare un'alternativa al *progetto JupyterHub*.

4.1 Requisiti del progetto

Si è stabilito che per essere un'alternativa valida, il progetto dovesse avere i seguenti requisiti:

- *estendibilità*
- *security by default*
- *interfaccia grafica facile ed intuitiva*
- *maggiore possibilità di collaborazione*
- *codice open source*
- *possibilità di personalizzazione*
- *accessibilità*

Seguono descrizioni dettagliate dei requisiti elencati.

4.1.1 Estendibilità

Aggiungere nuove funzionalità al software deve essere facile, e non richiedere ristrutturazioni profonde del codice.

Inoltre, il software deve essere **modulare**, in modo da semplificare l'aggiornamento, la sostituzione e la eventuale rimozione di componenti.

Infine, il software deve esporre un'**interfaccia alla quale altri software esterni possono connettersi** per interagirvi come se fossero un utente.

4.1.2 Sicurezza

I dati immagazzinati all'interno del software devono essere **protetti da acccessi non autorizzati**.

Tentativi di ingannare gli utenti del software devono essere impediti, riducendo il fattore umano nelle falle di sicurezza.

Non si reputa importante impedire agli utenti di comunicare con Internet all'interno delle loro ricerche, in quanto si ritiene che essi siano utenti fidati; qualora ne sorga la necessità, ciò deve essere possibile senza ristrutturazione del codice.

Non si reputa nemmeno importante limitare le risorse utilizzate dai *notebook* in uso; deve però essere possibile implementare la funzionalità in futuro, se divenisse necessario.

4.1.3 Intuitività

Il modo in cui utilizzare l'interfaccia utente del software deve essere **intuitiva** per l'utente medio, senza che egli abbia bisogno di leggere alcuna guida o manuale.

A tale scopo, l'interfaccia grafica deve utilizzare **design patterns comuni e familiari** all'utente medio.

In aggiunta, i **dettagli implementativi devono essere nascosti** all'utente, in modo che possa concentrarsi sull'utilizzo del software.

4.1.4 Personalizzabilità

Il software deve permettere all'utente di **personalizzare il suo workflow senza alcuna limitazione**, che ciò venga fatto tramite plugin, configurazioni speciali o modifica di file dell'ambiente di lavoro, assicurando che i workflow personalizzati di un utente **non possano interferire** con quelli degli altri.

Inoltre, il software deve inoltre permettere all'amministratore di **personalizzare nome e aspetto** mostrati agli utenti nell'interfaccia grafica, in modo che essa possa essere adattata al brand dell'istituzione che utilizza il progetto.

4.1.5 Possibilità di collaborazione

Il software deve permettere agli utenti di **collaborare sui notebook in tempo reale**, come all'interno dei *web-based editor*.

Devono essere **facilitate le interazioni tra utenti**, al fine di ridurre errori e incomprensioni tra essi.

4.1.6 Open source

Il software deve essere interamente **open source**.

In pieno spirito collaborativo, il **codice sorgente deve essere liberamente consultabile, modificabile, utilizzabile e condivisibile**, sia per soddisfare la curiosità degli utenti, sia per permetterne lo studio e il miglioramento.

Tutte le **modifiche al codice sorgente devono essere rese disponibili agli utenti** del software modificato, in modo che possano verificare l'affidabilità del software che utilizzano.

4.1.7 Responsività

Il software deve essere **utilizzabile su schermi di dimensione ridotta**, come quelli di un cellulare.

Pertanto, gli elementi dell'interfaccia devono essere disposti in modo tale da permetterne la visualizzazione corretta su schermi di qualsiasi dimensione e risoluzione.

4.1.8 Accessibilità

Il software deve essere utilizzabile da **qualsiasi tipologia di utente**, inclusi utenti con disabilità visive e motorie.

Deve essere quindi possibile utilizzare il software **interamente da tastiera**, senza dover ricorrere a un mouse.

Inoltre, i colori dell'interfaccia grafica devono **essere scelti favorendo l'accessibilità degli utenti daltonici**.

4.2 Separazione in moduli

Per realizzare il requisito dell'*estendibilità*, si è scelto di separare le parti dell'applicazioni in 4 diversi moduli interagenti.

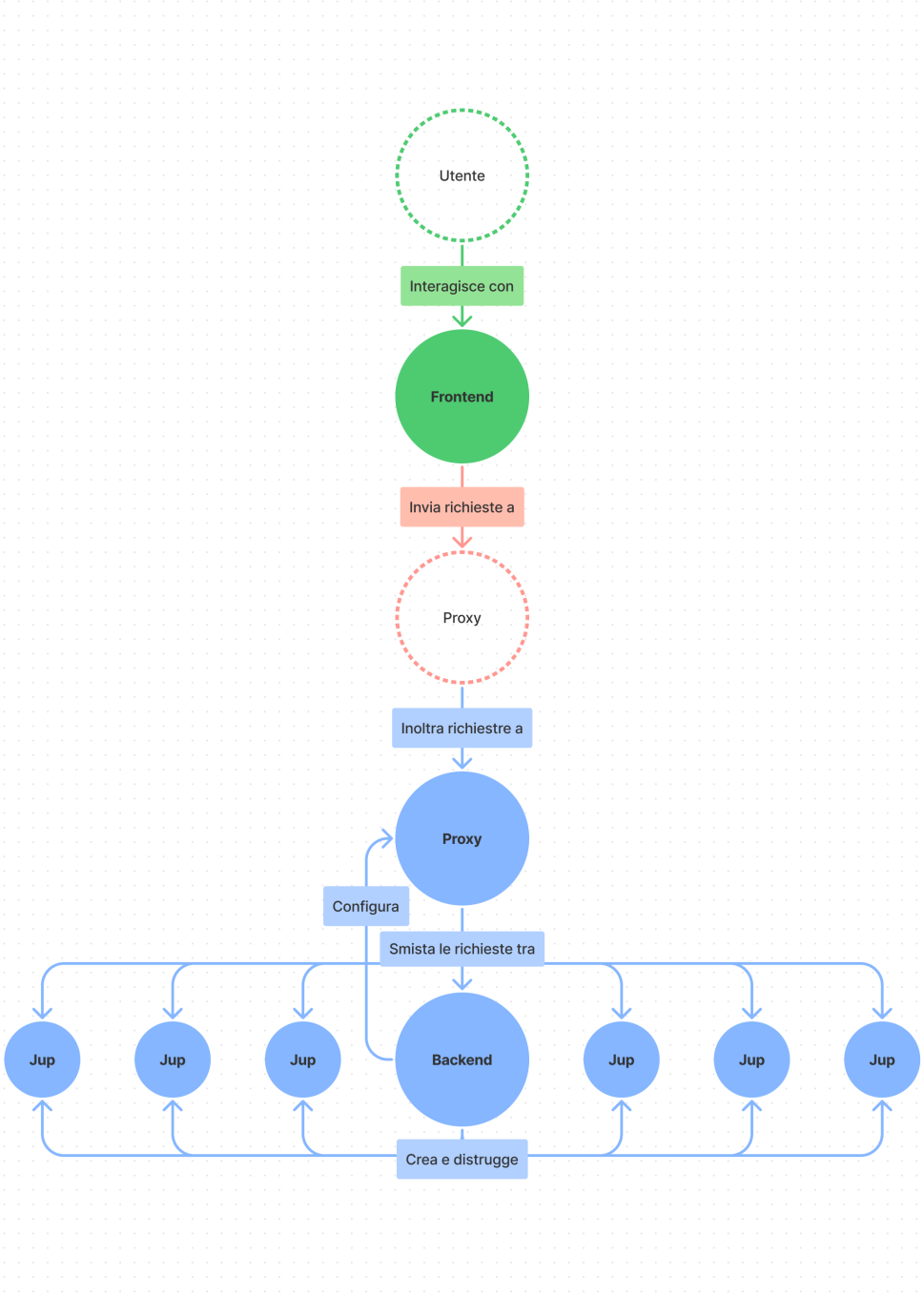


Figura 4.2.1: Schema che mostra come interagiscono tra loro i moduli di Sophon.

4.2.1 Modulo backend

Il modulo backend consiste in una web API (application programming interface) che si interfaccia con il database e i moduli Jupyter, permettendo un accesso controllato alle risorse del software.

È scritto in *Python*, usando *Poetry* e le librerie *Django*, *Django REST Framework* e *Docker SDK for Python*, descritte nei prossimi paragrafi.

Esso è **eseguito dal server** sul quale è ospitato Sophon.

4.2.1.1 Python

Python[21] è un linguaggio di programmazione orientato agli oggetti interpretato con tipizzazione dinamica forte, particolarmente popolare negli ambiti dello sviluppo web e data science.

Ha numerosissime librerie (dette *packages*), sia incluse nella distribuzione base del linguaggio, sia disponibili per il download sul Python Package Index[22].

La sua sintassi è semplice ed human-friendly, come è possibile vedere dal seguente frammento di codice:

```
class Animale:
   def verso():
      raise NotImplementedError()

class Cane(Animale):
   def verso():
      print("Woof!")

class Gatto(Animale):
   def verso():
      print("Miao!")

zoo = [
   Cane(),
   Gatto(),
   Cane(),
]

for animale in zoo:
   animale.verso()
```

La sua semplicità e l'enorme quantità di librerie a disposizione lo ha reso il secondo linguaggio di programmazione più popolare al mondo [so:survey2021], subito dopo *JavaScript*; proprio per questi motivi è stato scelto per lo sviluppo del modulo backend.

[21] https://www.python.org/
[22] https://pypi.org/

4.2.1.2 Poetry

Per gestire le dipendenze di Sophon si è scelto di usare Poetry[23], un innovativo package manager per il linguaggio Python.

Poetry è in grado di risolvere automaticamente alberi complessi di dipendenze, generando un *lockfile* (`poetry.lock`) con la soluzione adottata, in modo che le dipendenze utilizzate siano congelate e uguali per tutti gli ambienti in cui deve essere sviluppato Sophon.

4.2.1.3 Django

Django[24] è un framework Python per lo sviluppo di siti web dinamici.

Fornisce una suite di strumenti che assistono nella creazione di siti di medie dimensioni, come un ORM (object-relational model) per i database, una pagina di amministrazione integrata per la gestione dei contenuti del sito e un sistema di moduli scollegabili detti "app".

Le pagine restituite vengono definite attraverso funzioni, dette *function-based views*, o attraverso classi, dette *class-based views*, che ricevono in input la richiesta effettuata dall'utente ed restituscono in output la risposta HTTP da inoltrargli.

È stato scelto per la realizzazione del modulo backend in quanto presentato al corso di Tecnologie web di Unimore, e in quanto contenente tutte le funzionalità necessarie per la realizzazione del progetto del sito.

4.2.1.4 Django REST Framework

Django REST Framework[25] è un'estensione per *Django* che aggiunge la possibilità di inserire REST (representational state transfer) API all'interno delle applicazioni Django.

Permette di definire metodi dell'API in modo molto simile alle views di Django: si vengono a creare le *function-based API views* se i metodi sono definiti attraverso funzioni, o le *class-based API views* se i metodi sono definiti attraverso classi.

Inoltre, permette la generazione automatica di metodi per l'interazione con certe entità del database, attraverso particolari classi dette *viewset*.

Come per Django, è stato scelto per lo sviluppo di Sophon in quanto è stato presentato al corso di Tecnologie web di Unimore, e perchè si è ritenuto che fosse l'opzione più semplice per realizzare una web API all'interno di Django.

[23] https://python-poetry.org/
[24] https://www.djangoproject.com/
[25] https://www.django-rest-framework.org/

4.2.1.5 Docker SDK for Python

Per interfacciarsi con i *moduli Jupyter*, si è deciso di utilizzare Docker SDK for Python[26], un client Python per l'interazione con il daemon *Docker*.

Vedi anche:

Containerizzazione, più avanti nel capitolo.

4.2.2 Modulo frontend

Il *modulo frontend* consiste in una applicazione web che consente agli utenti di interagire con Sophon da un'interfaccia grafica.

Le interazioni vengono inviate al *modulo proxy*, che le ispeziona e le inoltra al *modulo backend*.

È scritto in *TypeScript*, usando *React* e le librerie *FontAwesome* e *Bluelib*, in aggiunta alle loro dipendenze ed altre piccole librerie di supporto.

Viene **eseguito dal browser web** dell'utente che desidera interagire con Sophon, transcompilato da TypeScript a *JavaScript*.

4.2.2.1 JavaScript

JavaScript[27] è un linguaggio di programmazione interpretato con tipizzazione dinamica debole.

È l'unico linguaggio utilizzabile per rendere interattive le pagine web; pertanto, è indirettamente utilizzato dal modulo frontend di Sophon.

Il suo modello di oggetti si basa su dizionari che mappano i nomi degli attributi ai loro corrispondenti valori.

Fa inoltre abbondante uso della capacità dei linguaggi dinamici di definire funzioni a runtime (dette anche callback), sfruttandole per favorire la programmazione funzionale.

```
const cane = {
   verso: () ⇒ console.log("Woof!"),
};

const gatto = {
   verso: () ⇒ console.log("Miao!"),
};

const zoo = [cane, gatto];

zoo.forEach(
   (animale) ⇒ animale.verso()
);
```

[26] https://docker-py.readthedocs.io/en/stable/
[27] https://it.wikipedia.org/wiki/JavaScript

4.2.2.2 Node.js

Node.js[28] è un runtime *JavaScript* che permette la scrittura e l'esecuzione di programmi all'esterno del contesto di un browser web, utilizzando invece come contesto il sistema operativo su cui viene eseguito.

Include NPM (Node package manager), un gestore di pacchetti per il download di librerie Node, che interagisce con l'npm Registry[29].

È utilizzato da Sophon come toolchain per lo sviluppo e il deployment del modulo frontend, in quanto necessario per l'esecuzione di *Create React App*.

4.2.2.3 Create React App

Create React App[30] è un insieme di strumenti *Node.js* per lo sviluppo di una applicazione web utilizzando la libreria per la creazione di interfacce grafiche *React*.

È utilizzato da Sophon per la costruzione della pagina del modulo frontend che sarà servita all'utente.

Si è scelto di usare Create React App in quanto astrae al programmatore tutta la logica di creazione della pagina, semplificando enormemente la manutenzione ed *estensione* futura del software.

4.2.2.4 TypeScript

TypeScript[31] è un'estensione al linguaggio di programmazione *JavaScript* che vi introduce un sistema di tipizzazione forte.

Non essendo immediatamente utilizzabile all'interno delle pagine web, deve essere prima convertito in JavaScript: ciò viene effettuato da *Create React App* in fase di costruzione dell'applicazione.

```
interface Animale {
   verso: () ⇒ string,
}

var cane: Animale = {
   verso: () ⇒ console.log("Woof!"),
};

var gatto: Animale = {
   verso: () ⇒ console.log("Miao!"),
};

var zoo: Animale[] = [cane, gatto];

zoo.forEach(
```

(continues on next page)

[28] https://nodejs.org/
[29] https://www.npmjs.com/
[30] https://create-react-app.dev/
[31] https://www.typescriptlang.org/

(continua dalla pagina precedente)

```
   (animale) ⇒ animale.verso()
);
```

È stata utilizzata in quasi ogni singola parte del modulo frontend, in quanto avere una tipizzazione forte riduce significativamente i bug prodotti e facilita manutenzione ed *estensione* del software.

4.2.2.5 React

React[32] è una libreria *JavaScript* per lo sviluppo di interfacce grafiche interattive all'interno di pagine web o applicazioni mobile.

L'interfaccia viene definita in modo dichiarativo e funzionale attraverso una variante dei linguaggi *JavaScript* (o *TypeScript*) detta JSX (o TSX), che permette l'inserimento di nodi HTML all'interno del codice.

Si basa sul concetto di *componenti*, piccole parti incapsulate di interfaccia grafica riutilizzabili attraverso tutta l'applicazione definite attraverso funzioni pure, e di *hooks*, particolari funzioni il cui nome inizia con `use` in grado di tenere traccia dello stato di un componente o di causare effetti collaterali all'interno di esso.

```
const ComponenteTitoloMaiuscolo = ({text}) ⇒ {
   const capitalizedText = text.toUpperCase();

   return (
      <h1>
         {capitalizedText}
      </h1>
   );
}
```

È stata scelta per l'utilizzo in Sophon in quanto permette la realizzazione di interfacce grafiche molto complesse attraverso codice di facile comprensione, rendendo possibile la creazione di un'interfaccia interattiva ed *intuitiva*.

4.2.2.6 FontAwesome

FontAwesome[33] è una libreria che fornisce più di mille icone utilizzabili gratuitamente all'interno di pagine web.

È stata usata per favorire l'*intuibilità* dell'interfaccia grafica attraverso simboli familiari all'utente.

[32] https://reactjs.org/
[33] https://fontawesome.com/

4.2.2.7 Bluelib

Bluelib[34] è un foglio di stile per pagine web orientato alla modularità, alla responsività e all'*accessibilità*.

È stato sviluppato nell'Estate 2021 come progetto personale dell'autore di questa tesi, ed è stato esteso con temi aggiuntivi in Autunno 2021, tra cui uno sviluppato appositamente per Sophon.

Si basa sul concetto di **pannelli**, sezioni di pagina separate dal resto tramite un colore di sfondo o un bordo diverso.

Fa ampio uso delle CSS Custom Properties[35], permettendo lo sviluppo di vari *temi* con aspetto differente.

Figura 4.2.2: Il tema "Royal Blue" (`royalblue`) di Bluelib, da cui ha origine il nome.

4.2.2.7.1 Bluelib React

Bluelib React[36] è un adattamento a *React* del foglio di stile *Bluelib*.

È stato sviluppato a inizio Autunno 2021 come parte del tirocinio interno dell'autore di questa tesi.

Definice componenti per ogni elemento grafico introdotto in Bluelib, e rende velocemente configurabili alcuni parametri, come il colore o la disabilitazione di un pannello.

[34] https://gh.steffo.eu/bluelib/
[35] https://developer.mozilla.org/en-US/docs/Web/CSS/--*
[36] http://gh.steffo.eu/bluelib-react/

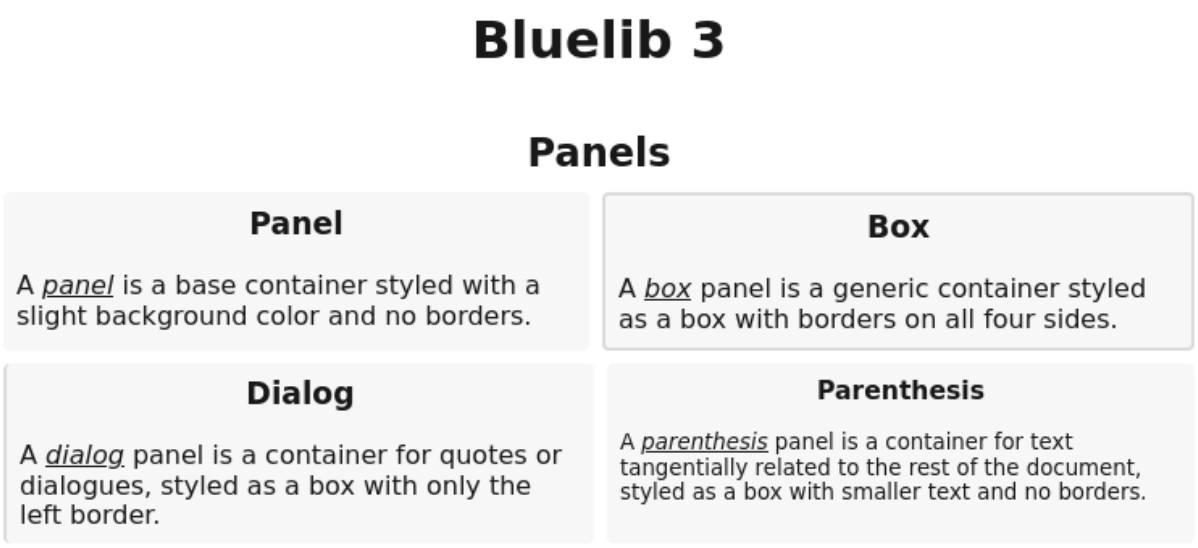


Figura 4.2.3: Il tema "Sheet of Paper" (`paper`) di Bluelib, pensato per la stampa su carta.

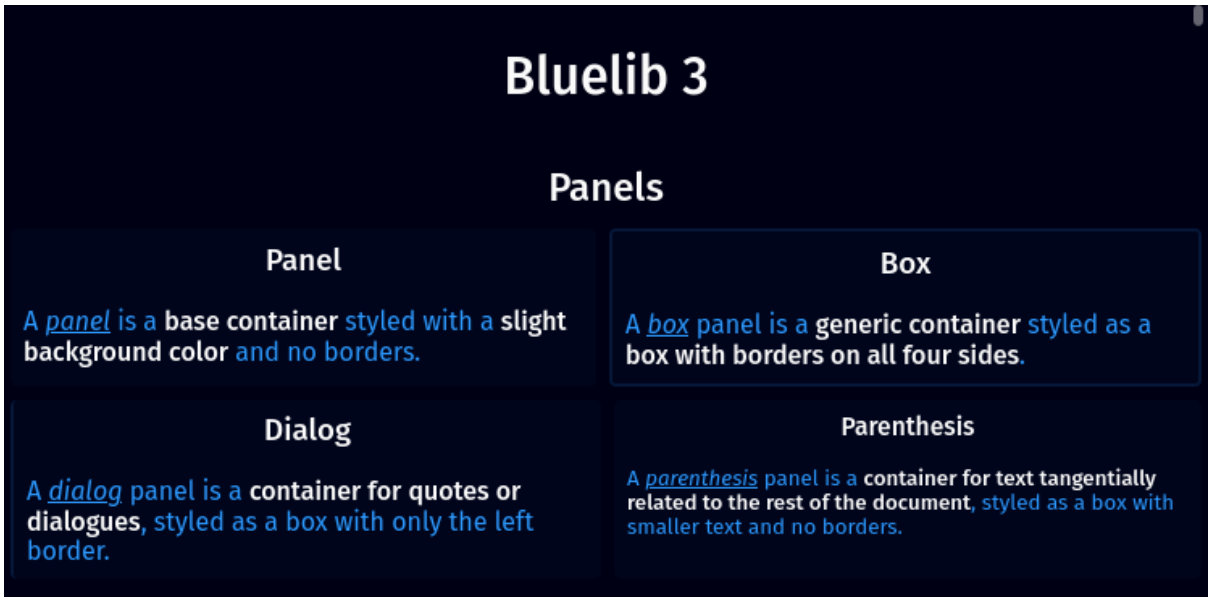


Figura 4.2.4: Il tema "The Sophonity" (`sophon`) di Bluelib, creato appositamente per questo progetto.

Figura 4.2.5: Il tema "Hacker Terminal" (`hacker`) di Bluelib, creato per testare la visualizzazione di caratteri monospace.

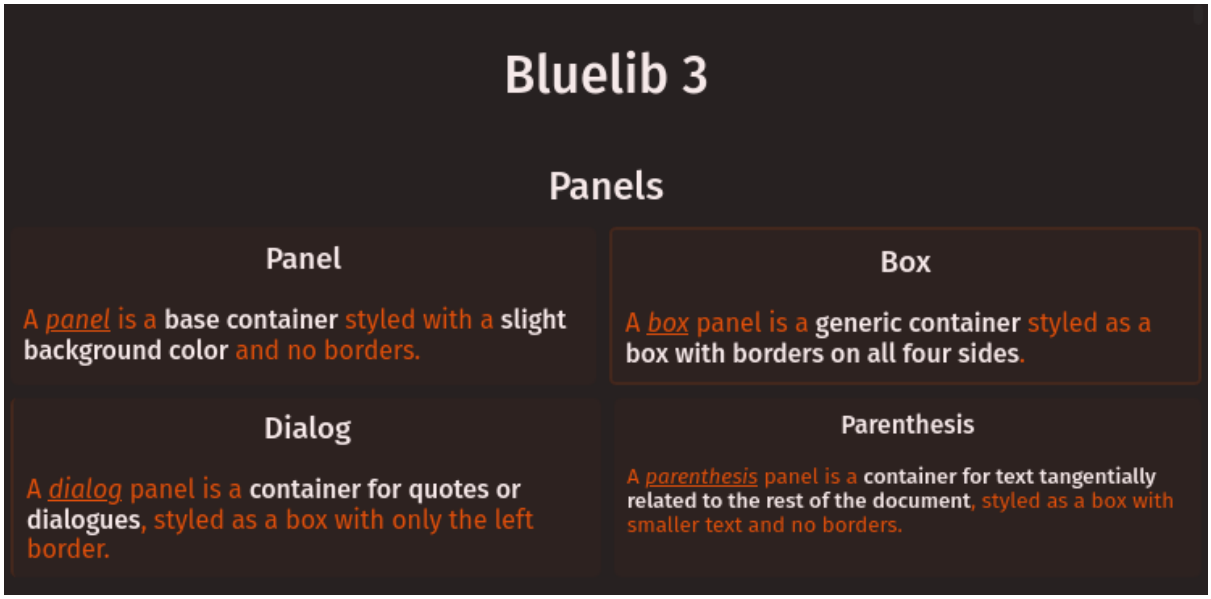


Figura 4.2.6: Il tema "Gestione Amber" (`amber`) di Bluelib, realizzato da Lorenzo Balugani.

4.2.3 Modulo proxy

Il *modulo proxy* consiste in un web server che permette di accedere al *modulo backend*, ai *moduli Jupyter* e a una versione preconfigurata del *modulo frontend*.

È stato realizzato configurando *Apache HTTP server* in modo che effettuasse dinamicamente *reverse proxying* verso gli altri moduli basandosi su una rubrica aggiornata dal backend.

Viene **eseguito dal server** sul quale è ospitato Sophon.

4.2.3.1 Reverse proxy

Il *reverse proxying* è un'operazione effettuabile dai web server per permettere l'accesso controllato ad altri web server collocati su una rete interna attraverso l'inoltro di pacchetti.

Frequentemente, il reverse proxying viene utilizzato per "aggiungere" l'HTTPS a un web server disponibile solo in HTTP, o per disambiguare tra più web server che devono essere accessibili allo stesso indirizzo IP ma con nomi di dominio diversi.

In un'installazione predefinita di Sophon, il reverse proxying effettuato è duplice:

- il server web della macchina host riceve richieste HTTPS e le inoltra in HTTP al server web del *modulo proxy*;
- il server web del modulo proxy riceve richieste HTTP che inoltra ai vari moduli in base al valore dell'header `Host` della richiesta ricevuta.

4.2.3.2 Apache HTTP server

Apache HTTP Server[37], comunemente chiamato anche *httpd* o *apache2*, è uno dei tre webserver "general purpose" più comunemente usati al mondo.

Ha una struttura a moduli, che forniscono funzionalità aggiuntive, ed è configurabile tramite uno o più file `.conf` aventi sintassi come la seguente:

```
# Questa è un'istruzione globale.
Bind 80

# Questo è un blocco di istruzioni ristretto a un contesto␣
↪specifico.
<VirtualHost *:80>
   ServerName "ilmiosophon.it"
   ServerAlias "*.ilmiosophon.it"

   RewriteEngine On
   RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>
```

[37] https://httpd.apache.org/

Figura 4.2.7: Schema del reverse proxying di Sophon.

4.2.4 Modulo Jupyter

Il *modulo Jupyter* consiste in una versione preconfigurata di *Jupyter* pronta per essere istanziata dal *modulo backend*.

Tanti moduli Jupyter possono esistere contemporaneamente su Sophon: ne viene creato uno per ogni *notebook computazionale* gestito dal modulo backend.

Viene **eseguito dal server** sul quale è ospitato Sophon.

4.3 Containerizzazione

Al fine di facilitare l'installazione e di migliorare la *sicurezza* dell'applicazione, si è stabilito di costruire *container Docker* per tutti i moduli di Sophon.

4.3.1 Docker

Docker[38] è un software che permette di eseguire applicazioni all'interno di *container* isolati dal resto del sistema, in maniera simile all'esecuzione di macchine virtuali, ma **condividendo il kernel** con la macchina host.

È composto da due parti, *Docker Engine* e *Docker Compose*, e prevede varie astrazioni, quali le *immagini*, i *container*, i *network* e i *volumi*.

4.3.1.1 Immagini Docker

Le *immagini* Docker sono sequenze di regole e insiemi di file per la creazione di un *container*, tipicamente partendo da un altro container come base. [docker:overview]

Utilizzano un filesystem copy-on-write a strati: vengono registrate all'interno dell'immagine solamente le modifiche che ogni regola ha apportato al filesystem interno, rendendo le immagini molto più leggere di quanto lo sarebbero se dovesse essere salvato tutto il disco virtuale.

Possono essere comparate a immagini di macchine virtuali con tanti "punti di ripristino".

4.3.1.2 Container Docker

I *container* Docker sono istanze di *immagini* che possono essere eseguite dal *Docker Engine* [docker:overview].

Sono l'equivalente di un'intera macchina virtuale, che può essere avviata o arrestata.

[38] https://www.docker.com/

4.3.1.3 Network Docker

I *network* Docker sono astrazioni per vari tipi di reti di calcolatori: in particolare, essi permettono di collegare vari *container* ad una rete locale virtuale, permettendone l'interazione [docker:networking].

All'interno di un network è disponibile una funzionalità di risoluzione automatica degli indirizzi IP virtuali dei container: per accedere al container `pear` in HTTP, ad esempio, sarà sufficiente utilizzare `apple` come se fosse un nome di dominio: `http://pear/`.

Sono una versione più elaborata ed efficiente dei moduli di rete per macchine virtuali.

4.3.1.4 Volumi Docker

I *volumi* Docker sono astrazioni per filesystem che permettono la permanenza e la condivisione tra container di file [docker:volumes].

Essi vengono montati all'interno di un container in una cartella configurabile detta *mount point*; tutti i container con accesso al volume vedranno gli stessi file all'interno di essa.

Sono il parallelo delle immagini disco delle macchine virtuali.

4.3.2 Docker Engine

Docker Engine[39] è il daemon che si occupa della gestione di *immagini*, *container*, *network* e *volumi*.

Astrae la piattaforma su cui viene eseguito, in modo che tutte le immagini possano essere eseguite su Linux come su Windows o Mac OS X.

4.3.3 Docker Compose

Docker Compose[40] è uno strumento da linea di comando che permette l'esecuzione di applicazioni Docker composte da più container.

Le applicazioni Compose sono definite all'interno di un file YAML[41] come il seguente:

```yaml
version: "3.9"

# Elenco dei volumi dell'applicazione
volumes:
  db-data:

# Elenco dei network dell'applicazione
networks:
  main:
```

(continues on next page)

[39] https://docs.docker.com/engine/
[40] https://docs.docker.com/compose/
[41] https://it.wikipedia.org/wiki/YAML

(continua dalla pagina precedente)

```
# Elenco dei container dell'applicazione
services:
  db:
    # Immagine del container
    image: postgres
    # Mount point dei volumi del container
    volumes:
      - db-data:/var/lib/postgresql/data
    # Network del container
    networks:
      - main

  app:
    image: my-app-image
    networks:
      - main
    # Container richiesti da questo container
    depends_on:
      - db
```

4.4 Controllo versione

Per assistere nello sviluppo del software si è deciso di utilizzare il sistema di controllo versione *Git* in ogni fase dello sviluppo del progetto.

Inoltre, per favorire lo sviluppo di una community *open source* attorno a Sophon, si è deciso di pubblicare il progetto su *GitHub*, sotto la *Affero General Public License 3.0+*.

4.4.1 Git

Git[42] è un software di controllo versione, ovvero un software in grado di tenere traccia di modifiche effettuate su file, in modo da mantenerne uno storico, e permettere a più autori di lavorare in parallelo su documenti.

Inizialmente realizzato da Linus Torvalds per lo sviluppo del kernel Linux, ha preso velocemente piede in tutto il settore dello sviluppo software, diventando di fatto lo standard per lo sviluppo collaborativo di software.

Le cartelle di file tracciate da Git sono dette *repository*, mentre un blocco atomico di modifiche è detto *commit*.

[42] https://git-scm.com/

4.4.2 GitHub

GitHub[43] è un servizio web di Microsoft per l'hosting e la pubblicazione di repository Git.

Per ciascun repository sono messe a disposizione gratuitamente numerose funzionalità, quali un issue tracker, strumenti di code review e sistemi di automazione per lo sviluppo [github:features].

4.4.3 Affero General Public License 3.0+

Sophon è rilasciato sotto la GNU Affero General Public License v3[44] (o successiva).

Il testo completo della licenza è disponibile all'interno del file LICENSE.txt[45] allegato al codice sorgente del software.

In breve, la licenza, detta virale, permette a chiunque di utilizzare, distribuire e modificare il software, a condizione che qualsiasi modifica venga ri-distribuita agli utenti del software modificato utilizzando la stessa licenza.

Si specifica che la licenza copre tutti i file all'interno del repository `Steffo99/sophon`, anche se essi non contengono un header che indica che sono protetti da copyright.

4.5 Entità di Sophon

Al fine di definire più in dettaglio le operazioni che devono poter essere effettuate all'interno di Sophon, sono state definite delle *entità*, i tipi base con cui l'utente può interagire.

4.5.1 Istanza in Sophon

Un'*istanza* rappresenta un'**installazione di Sophon** effettuata su un server di un'istituzione di ricerca, come ad esempio un'Università.

Ogni istanza è **fisicamente e logicamente separata** dalle altre; istanze diverse **non condividono alcun dato** tra loro.

4.5.1.1 URL dell'istanza

Ciascuna istanza è accessibile tramite **uno specifico URL**, scelto dall'amministratore di sistema al momento dell'installazione.

[43] https://github.com/
[44] https://www.gnu.org/licenses/agpl-3.0.html
[45] https://github.com/Steffo99/sophon/blob/main/LICENSE.txt

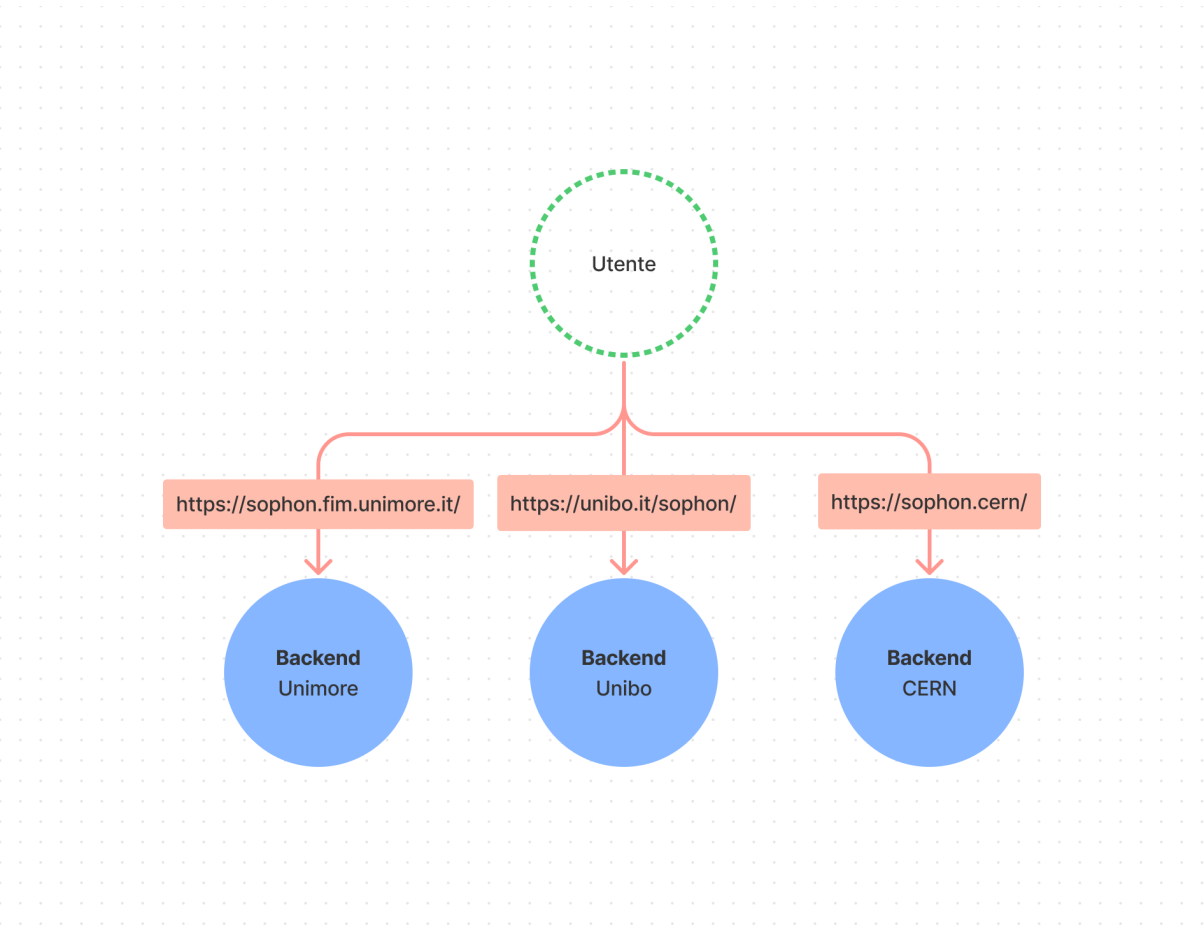


Figura 4.5.1: Schema rappresentante un esempio di URL di istanza rispettivamente per Unimore, Unibo e il CERN. Si noti come Sophon possa essere ospitato a domini di qualsiasi livello o radici diverse da /, quella predefinita.

4.5.2 Utenti in Sophon

Un *utente* è una entità che interagisce con una specifica istanza Sophon: ad esempio, un utente potrebbe essere una persona fisica, oppure potrebbe essere un software di automazione che si interfaccia con Sophon.

La tabella viene creata automaticamente da Django all'interno di ogni applicazione che include

4.5.2.1 Livelli di accesso

Un utente può avere uno dei seguenti *livelli di accesso*:

Superutente Utente con accesso completo a ogni singola risorsa sull'istanza Sophon, tipicamente riservato per l'amministratore di sistema.

Utente Utente con permessi limitati alle risorse che ha creato o a cui è stato fornito accesso.

Ospite Utente che può visualizzare alcuni contenuti dell'istanza Sophon ma non può interagirci.

4.5.2.2 Credenziali di accesso

Gli utenti di tipo *Utente* e *Superutente* devono identificarsi sull'istanza con le loro credenziali.

Di default, le credenziali sono un **nome utente** e una **password**, ma è possibile implementare un sistema diverso, ad esempio un sistema SSO (Single Sign-On).

4.5.3 Gruppi di ricerca in Sophon

Un *gruppo di ricerca* rappresenta un insieme di utenti che collaborano su uno o più progetti.

4.5.3.1 Membri e modalità di accesso

Gli utenti dell'*istanza* possono diventare *membri* dei gruppi di ricerca, con una delle seguenti modalità selezionate nelle impostazioni del gruppo:

- se il gruppo è *aperto*, allora qualsiasi utente potrà diventarne membro semplicemente **facendo richiesta** attraverso l'interfaccia web;
- se il gruppo è in *modalità manuale*, allora nessun utente potrà richiedere di unirsi, e i membri saranno **selezionati manualmente** dal creatore del gruppo.

In qualsiasi momento, i membri di un gruppo possono **lasciarlo** facendo apposita richiesta attraverso il frontend.

4.5.3.2 Creazione di nuovi gruppi

Qualsiasi *utente* può **creare** gruppi di ricerca dall'interfaccia web.

4.5.3.3 Modifica di gruppi

Il creatore di un gruppo di ricerca è l'unico *utente* che può cambiarne **nome**, **descrizione**, **membri** e **modalità di accesso**.

Lo *slug*, l'identificatore univoco del gruppo, non è modificabile successivamente alla creazione, in quanto verrà utilizzato all'interno degli URL, che devono essere immutabili.

4.5.3.4 Eliminazione di gruppi

Il creatore di un gruppo è l'unico utente in grado di **cancellare** il gruppo che ha creato.

> **Pericolo:** L'eliminazione di un gruppo è un'operazione distruttiva non reversibile!

4.5.4 Progetti di ricerca in Sophon

Un *progetto di ricerca* rappresenta una **collezione di oggetti** relativa a un singolo argomento mantenuta da un *gruppo di ricerca*.

4.5.4.1 Visibilità dei progetti

I progetti hanno tre diverse impostazioni di visibilità che regolano chi può visualizzarne i contenuti:

Progetto privato Il progetto è visibile **solo ai membri del gruppo** a cui appartiene il progetto.

Progetto interno Il progetto è visibile **solo agli utenti** dell'istanza, e non agli ospiti.

Progetto pubblico Il progetto è visibile **a tutti**.

4.5.4.2 Creazione di nuovi progetti

Qualsiasi *membro* di un *gruppo di ricerca* può creare nuovi progetti.

4.5.4.3 Modifica di progetti

Qualsiasi *membro* di un *gruppo di ricerca* può modificare **nome**, **descrizione** dei progetti al suo interno.

Solo il *creatore del gruppo* può modificarne la **visibilità**, o **trasferire il progetto ad un altro gruppo**.

Lo *slug*, l'identificatore univoco del progetto, non è modificabile successivamente alla creazione, in quanto è utilizzato all'interno degli URL, che devono essere immutabili.

4.5.4.4 Eliminazione di progetti

Qualsiasi *membro* di un *gruppo di ricerca* può eliminare i progetti al suo interno.

> **Pericolo:** L'eliminazione di un progetto è un'operazione distruttiva non reversibile!

4.5.5 Notebook in Sophon

Un *notebook* rappresenta una **postazione di lavoro** che può essere allegata ad un *progetto di ricerca*.

4.5.5.1 Creazione di nuovi notebook

Qualsiasi **membro** di un *gruppo di ricerca* può creare nuovi notebook all'interno di uno dei progetti del gruppo a cui appartiene.

4.5.5.2 Slug riservati

Un notebook non può avere come *slug* uno dei seguenti valori, in quanto riservati per altri usi:

- `backend`
- `frontend`
- `proxy`
- `api`
- `static`
- `src`

In più, uno slug di un notebook non può iniziare o terminare con un trattino `-`, in quanto risulterebbe in un sottodominio non valido.

4.5.5.3 Stato del notebook

Un notebook può essere *avviato* o *fermo* in base al suo stato di esecuzione sull'*istanza* Sophon:

- è *avviato* se è in esecuzione e accessibile;
- è *fermo* se non è in esecuzione o se è in fase di preparazione.

Alla creazione, un notebook è *fermo*.

4.5.5.3.1 Avviare un notebook

Un **membro** del *gruppo di ricerca* a cui appartiene il notebook può richiedere al server l'avvio di quest'ultimo, in modo da poterlo utilizzare successivamente.

4.5.5.3.2 Fermare un notebook

Un **membro** del *gruppo di ricerca* a cui appartiene il notebook può richiedere al server l'arresto di quest'ultimo, salvando i dati e interrompendo la sessione di lavoro attualmente in corso.

4.5.5.4 Immagine del notebook

In **fase di creazione** di un notebook, oppure mentre esso è **fermo**, è possibile selezionare l'*immagine Docker* che esso deve eseguire all'avvio.

Di default, l'immagine deve essere quella del *modulo Jupyter*.

Le immagini ammesse devono esporre un server HTTP sulla porta 8080, su cui verrà fatto *reverse proxying* dal *modulo proxy*.

4.5.5.5 Collegamento a un notebook

I **membri** del *gruppo di ricerca* a cui appartiene il notebook possono connettersi ad un notebook **avviato** attraverso un URL segreto comunicatogli dal *modulo backend*.

L'URL segreto è ottenuto inserendo come query parameter dell'URL del notebook il token di autenticazione di *Jupyter*.

4.5.5.6 Blocco di un notebook

Qualsiasi **membro** del *gruppo di ricerca* a cui appartiene il notebook può *bloccarlo* per segnalare agli altri utenti che vi hanno accesso di non utilizzare quello specifico notebook.

Bloccare un notebook **rimuove dall'interfaccia web** i bottoni di interazione con esso per tutti gli utenti, tranne l'utente richiedente il blocco.

Nota: Il blocco di un notebook **è solo estetico**, e non ha lo scopo di impedire agli utenti di interagire con il notebook, ma serve per indicare ai propri collaboratori che si stanno effettuando modifiche che non permettono collaborazione sul notebook.

Un notebook bloccato può essere sbloccato da qualsiasi **membro** del *gruppo di ricerca*; il membro che ha richiesto il blocco potrà sbloccarlo **immediatamente**, mentre agli altri membri è richiesto di confermare l'azione.

4.5.5.7 Modifica di un notebook

Qualsiasi *membro* di un *gruppo di ricerca* può modificare **nome** e **immagine** dei notebook *fermi* al suo interno.

I notebook *avviati* non possono essere modificati.

Lo *slug*, l'identificatore univoco del notebook, non è modificabile successivamente alla creazione, in quanto è utilizzato all'interno degli URL, che devono essere immutabili.

4.5.5.8 Eliminazione di un notebook

Qualsiasi *membro* di un *gruppo di ricerca* può eliminare i notebook all'interno dei progetti del gruppo, a condizione che questi siano *fermi* e *non bloccati*.

4.6 Database

Il *modulo backend* di Sophon necessita di archiviare dati persistenti altamente relazionali; pertanto, è stato necessario adottare una soluzione in grado di gestirli.

A tale scopo, è stato selezionato il database relazionale PostgreSQL[46], in quanto FLOSS (Free and Libre Open Source Software), adatto a dati relazionali, compatibile con Django, e ampiamente utilizzato in tutto il mondo.

[46] https://www.postgresql.org/

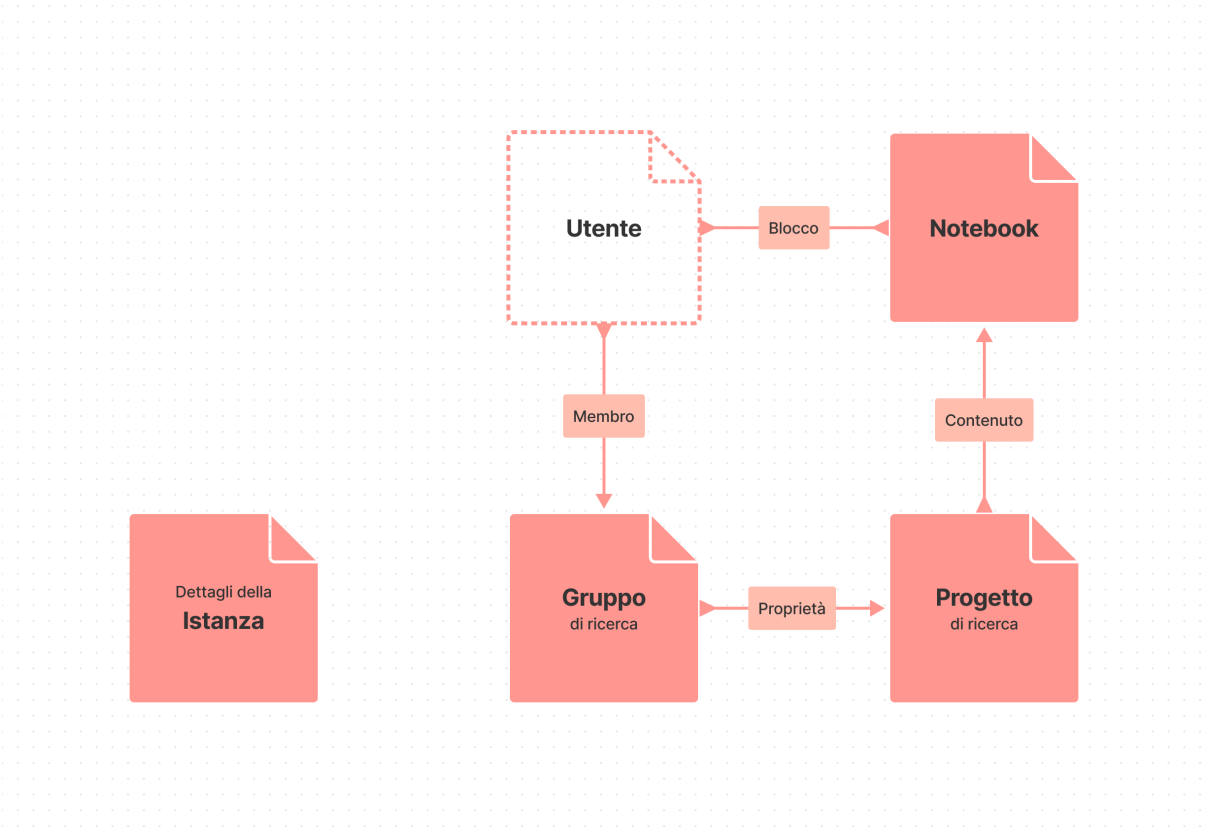


Figura 4.6.1: Schema semplificato del database di Sophon.

CAPITOLO 5

Realizzazione di Sophon

Terminato il progetto, si è passati a realizzarne una versione funzionante su calcolatore.

5.1 Realizzazione del modulo backend

Il modulo backend è stato realizzato come un package *Python* denominato `sophon`, e poi *containerizzato*, creando un'immagine *Docker* standalone.

5.1.1 Il project Django

Il package è stato creato utilizzando l'utility `startproject` di Django, la quale crea una cartella di script *Python* con i quali partire per lo sviluppo di una nuovo software web.

La cartella generata è stata modificata significativamente: ne si è modificata la struttura in modo tale da trasformarla da un insieme di script a un vero e proprio modulo Python eseguibile e distribuibile, e si sono aggiunte nuove funzionalità di utilità generale all'applicazione, quali una *app di amministrazione personalizzata*, il *caricamento dinamico delle impostazioni* e vari *miglioramenti all'autenticazione*

5.1.1.1 App di amministrazione personalizzata

L'app di amministrazione di Django `django.contrib.admin`[47] viene personalizzata con la classe `SophonAdminSite`, che ne modifica alcuni parametri.

Inoltre, il template predefinito viene sovrascritto dal file `templates/admin/base.html`, che sostituisce il foglio di stile con uno personalizzato per Sophon.

`class SophonAdminSite`(*django.contrib.admin.AdminSite*)

`site_header = "Sophon Server Administration"`
Il nome della pagina nell'header viene modificato a *Sophon Server Administration*.

`site_title = "Sophon Server Administration"`
Il titolo della pagina nell'header viene anch'esso modificato a *Sophon Server Administration*.

`site_url = None`
Il collegamento *View Site* viene rimosso, in quanto è possibile accedere all'interfaccia web di Sophon da più domini contemporaneamente.

`index_title = "Resources Administration"`
Il titolo dell'indice viene modificato a *Resources Administration*.

`class SophonAdminConfig`(*django.contrib.admin.apps.AdminConfig*)

`default_site = "sophon.admin.SophonAdminSite"`
`SophonAdminSite` è selezionata come classe predefinita per il sito di amministrazione.

5.1.1.2 Caricamento dinamico delle impostazioni

Il file di impostazioni viene modificato per **permettere la configurazione attraverso variabili di ambiente** invece che attraverso la modifica del file `settings.py`, rendendo la *containerizzazione* molto più semplice.

```
try:
    DATABASE_ENGINE = os.environ["DJANGO_DATABASE_ENGINE"]
except KeyError:
    log.warning("DJANGO_DATABASE_ENGINE was not set, defaulting to_
 ↪PostgreSQL")
    DATABASE_ENGINE = "django.db.backends.postgresql"
log.debug(f"{DATABASE_ENGINE = }")
```

Inoltre, viene configurato il modulo `logging`[48] per emettere testo colorato di più facile comprensione usando il package `coloredlogs`[49].

[47] http://docs.djangoproject.com/en/3.2/ref/contrib/admin/#module-django.contrib.admin
[48] https://docs.python.org/3.8/library/logging.html#module-logging
[49] https://coloredlogs.readthedocs.io/en/latest/api.html#module-coloredlogs

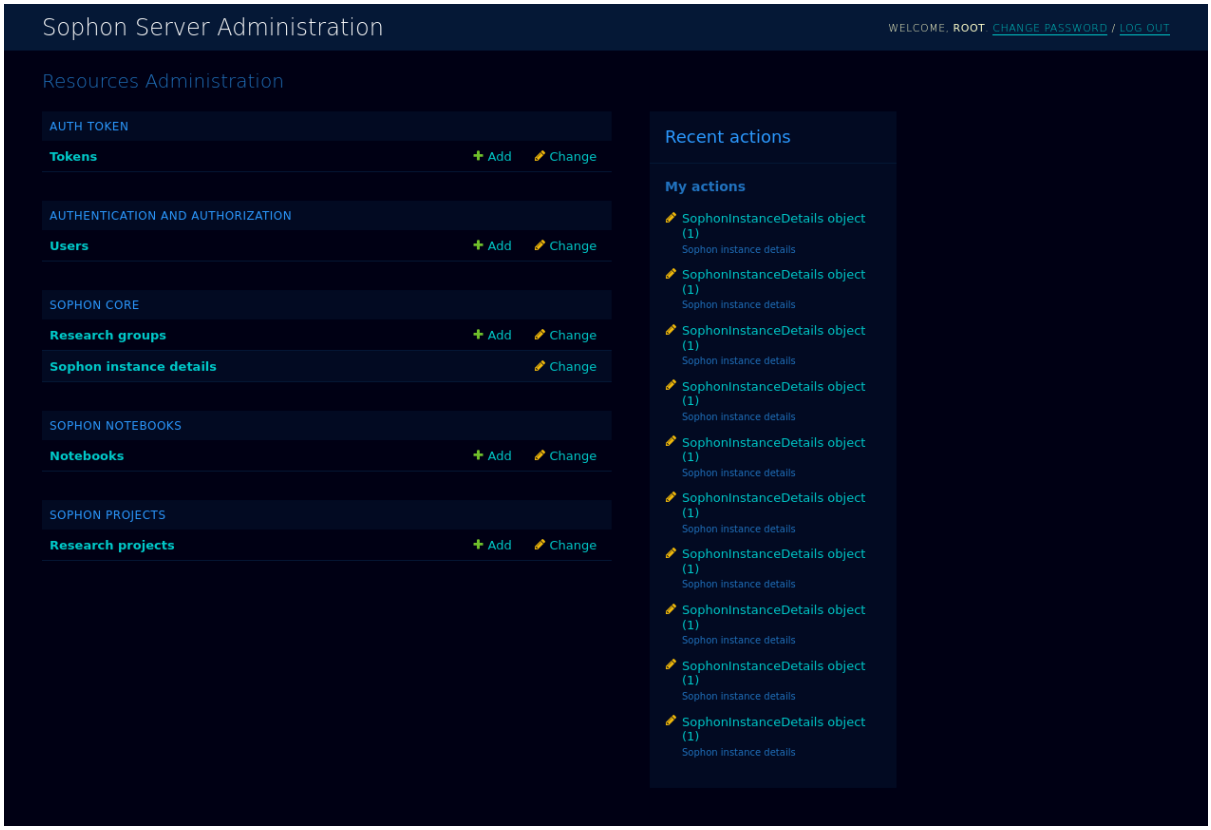


Figura 5.1.1: Immagine della pagina principale dell'app di amministrazione.

```
"detail": {
    "()": coloredlogs.ColoredFormatter,
    "format": "{asctime:>19} | {name:<24} | {levelname:>8} |
↪{message}",
    "style": "{",
}
```

Una lista di tutte le variabili di ambiente di configurazione è riportata nel capitolo *Installazione di Sophon*.

5.1.1.3 Miglioramenti all'autenticazione

La classe `rest_framework.authentication.TokenAuthentication` viene modificata per ottenere un comportamento conforme allo standard della Bearer authentication.

`class BearerTokenAuthentication`(*rest_framework.authentication.TokenAuthentication*)

`keyword = "Bearer"`

Si configura `rest_framework` per accettare header di autenticazione nella forma `Bearer <token>`, invece che il default di `rest_framework Token <token>`.

La view `rest_framework.authtoken.views.ObtainAuthToken` viene estesa per aggiungere dati alla risposta di autenticazione riuscita.

class CustomObtainAuthToken(*rest_framework.authtoken.views.ObtainAuthToken*)

post(*self*, *request*, **args*, ***kwargs*)

In particolare, viene aggiunta una chiave `user`, che contiene i dettagli sull'utente che ha effettuato il login.

5.1.2 L'app Sophon Core

L'app `sophon.core` è l'app principale del progetto, e non può essere disattivata, in quanto dipendenza obbligatoria di tutte le altre app.

5.1.2.1 Aggiunta di un nuovo comando di gestione

Per permettere l'integrazione la creazione automatica del primo superutente quando Sophon viene eseguito da Docker, viene introdotto dall'app il comando di gestione `initsuperuser`.

class Command

Questo comando crea automaticamente un superutente con le credenziali specificate in `DJANGO_SU_USERNAME`, `DJANGO_SU_EMAIL` e `DJANGO_SU_PASSWORD`.

5.1.2.2 Modello base astratto

Viene estesa la classe astratta `django.db.models.Model`[50] con funzioni per stabilire il *livello di accesso* di un *utente* all'oggetto e per generare automaticamente i `rest_framework.serializers.ModelSerializer` in base ad esso.

class SophonModel(*django.db.models.Model*)

abstract can_edit(*self*, *user: django.contrib.auth.models.User*[51]) → bool[52]

Controlla se un utente può modificare l'oggetto attuale.

Parametri **`user`** -- L'utente da controllare.

Ritorna `True`[53] se l'utente deve poter modificare l'oggetto, altrimenti `False`[54].

abstract can_admin(*self*, *user: django.contrib.auth.models.User*[55]) → bool[56]

Controlla se un utente può amministrare l'oggetto attuale.

Parametri **`user`** -- L'utente da controllare.

Ritorna `True`[57] se l'utente deve poter amministrare l'oggetto, altrimenti `False`[58].

[50] http://docs.djangoproject.com/en/3.2/ref/models/instances/#django.db.models.Model

classmethod get_fields(*cls*) → set[59][str[60]]

Ritorna il `set`[61] di nomi di campi che devono essere mostrati quando viene richiesto l'oggetto attraverso l'API.

classmethod get_editable_fields(*cls*) → set[62][str[63]]

Ritorna il `set`[64] di nomi di campi di cui deve essere permessa la modifica se l'utente può modificare (`can_edit()`) l'oggetto.

classmethod get_administrable_fields(*cls*) → set[65][str[66]]

Ritorna il `set`[67] di nomi di campi di cui deve essere permessa la modifica se l'utente può amministrare (`can_admin()`) l'oggetto.

classmethod get_creation_fields(*cls*) → set[68][str[69]]

Ritorna il `set`[70] di nomi di campi che possono essere specificati dall'utente al momento della creazione dell'oggetto.

5.1.2.3 Modello di autorizzazione astratto

Viene definito un nuovo modello astratto, basato su `SophonModel`, che permette di determinare i permessi dell'*utente* in base alla sua appartenenza al gruppo a cui è collegato l'oggetto implementatore.

class SophonGroupModel(*SophonModel*)

abstract get_group(*self*) → *ResearchGroup*

[51] http://docs.djangoproject.com/en/3.2/ref/contrib/auth/#django.contrib.auth.models.User
[52] https://docs.python.org/3.8/library/functions.html#bool
[53] https://docs.python.org/3.8/library/constants.html#True
[54] https://docs.python.org/3.8/library/constants.html#False
[55] http://docs.djangoproject.com/en/3.2/ref/contrib/auth/#django.contrib.auth.models.User
[56] https://docs.python.org/3.8/library/functions.html#bool
[57] https://docs.python.org/3.8/library/constants.html#True
[58] https://docs.python.org/3.8/library/constants.html#False
[59] https://docs.python.org/3.8/library/stdtypes.html#set
[60] https://docs.python.org/3.8/library/stdtypes.html#str
[61] https://docs.python.org/3.8/library/stdtypes.html#set
[62] https://docs.python.org/3.8/library/stdtypes.html#set
[63] https://docs.python.org/3.8/library/stdtypes.html#str
[64] https://docs.python.org/3.8/library/stdtypes.html#set
[65] https://docs.python.org/3.8/library/stdtypes.html#set
[66] https://docs.python.org/3.8/library/stdtypes.html#str
[67] https://docs.python.org/3.8/library/stdtypes.html#set
[68] https://docs.python.org/3.8/library/stdtypes.html#set
[69] https://docs.python.org/3.8/library/stdtypes.html#str
[70] https://docs.python.org/3.8/library/stdtypes.html#set

Ritorna Il gruppo a cui appartiene l'oggetto.

classmethod get_access_to_edit(*cls*) → *sophon.core.enums.SophonGroupAccess*

Ritorna Il livello di autorità all'interno del gruppo necessario per modificare l'oggetto.

classmethod get_access_to_admin(*cls*) → *sophon.core.enums.SophonGroupAccess*

Ritorna Il livello di autorità all'interno del gruppo necessario per amministrare l'oggetto.

get_access_serializer(*self*, *user: User*) → typing.Type[71][rest_framework.serializers.ModelSerializer]

Ritorna Restituisce il `rest_framework.serializers.ModelSerializer` adeguato al livello di autorità dell'utente.

class sophon.core.enums.**SophonGroupAccess**(*enum.IntEnum*)

Enumerazione che stabilisce il livello di autorità che un *utente* può avere all'interno di un *gruppo di ricerca*.

NONE = 0

Ospite.

REGISTERED = 10

Utente registrato.

MEMBER = 50

Membro del gruppo al quale appartiene l'oggetto.

OWNER = 100

Creatore del gruppo al quale appartiene l'oggetto.

SUPERUSER = 200

Superutente con privilegi universali.

5.1.2.4 Modello dei dettagli dell'istanza

Viene creato il modello che rappresenta i dettagli dell'*istanza di Sophon*.

class SophonInstanceDetails(*SophonModel*)

id: IntegerField [1]

Impostando `1` come unica scelta per il campo della chiave primaria `id`, si crea un modello "singleton", ovvero un modello di cui può esistere un'istanza sola in tutto il database.

[71] https://docs.python.org/3.8/library/typing.html#typing.Type

L'istanza unica viene creata dalla migrazione `0004_sophoninstancedetails.py`.

name: CharField
Il titolo dell'istanza Sophon.

description: TextField
La descrizione dell'istanza Sophon, da visualizzare in un riquadro "A proposito dell'istanza".

theme: CharField ["sophon", "paper", "royalblue", "hacker", "amber"]
Il tema *Bluelib* dell'istanza.

version: str

Ritorna La versione installata del pacchetto `sophon`.

5.1.2.5 Modello del gruppo di ricerca

Viene creato il modello che rappresenta un *gruppo di ricerca*.

class ResearchGroup(*SophonGroupModel*)

slug: SlugField
L'identificatore del gruppo di ricerca, usato nei percorsi dell'API.

name: CharField
Il nome del gruppo di ricerca.

description: TextField
La descrizione del gruppo di ricerca, da visualizzare in un riquadro "A proposito del gruppo".

members: ManyToManyField → django.contrib.auth.models.User
Elenco dei membri del gruppo. L'utente `owner` è ignorato, in quanto è considerato sempre parte del gruppo.

owner: ForeignKey → django.contrib.auth.models.User
Il creatore e proprietario del gruppo, con privilegi amministrativi.

access: CharField ["MANUAL", "OPEN"]
La *modalità di accesso* del gruppo.

5.1.2.6 Estensione ai permessi di Django

I permessi di `rest_framework` vengono estesi con due nuove classi che utilizzano il *modello di autorizzazione astratto* precedentemente definito.

class Edit(*rest_framework.permissions.BasePermission*)
Consente l'interazione solo agli utenti che possono modificare l'oggetto.

class Admin(*rest_framework.permissions.BasePermission*)
Consente l'interazione solo agli utenti che possono amministrare l'oggetto.

5.1.2.7 Viewset astratti

Vengono definiti tre viewset in grado di utilizzare i metodi aggiunti dalle classi astratte `models.SophonModel` e `models.SophonGroupModel`.

class ReadSophonViewSet(*rest_framework.viewsets.ReadOnlyModelViewSet*, *metaclass=abc.ABCMeta*)
Classe **astratta** che estende la classe base `rest_framework.viewsets.ReadOnlyModelViewSet` con metodi di utilità mancanti nell'implementazione originale, allacciandola inoltre a `models.SophonGroupModel`.

abstract get_queryset(*self*) → QuerySet
Imposta come astratto (e quindi obbligatorio) il metodo `rest_framework.viewsets.ReadOnlyModelViewSet.get_queryset()`.

property permission_classes(*self*)
Sovrascrive il campo di classe `rest_framework.viewsets.ReadOnlyModelViewSet.permission_classes` con una funzione, permettendone la selezione dei permessi richiesti al momento di ricezione di una richiesta HTTP (invece che al momento di definizione della classe).

Delega la selezione delle classi a `get_permission_classes()`.

get_permission_classes(*self*) → typing.Collection[72][typing.Type[73][permissions.BasePermission]]
Funzione che permette la selezione dei permessi necessari per effetuare una determinata richiesta al momento di ricezione di quest'ultima.

Utile per le classi che erediteranno da questa.

get_serializer_class(*self*) → typing.Type[74][Serializer]
Funzione che permette la selezione del `rest_framework.serializers.Serializer` da utilizzare per una determinata richiesta al momento di ricezione di quest'ultima.

Utilizza:

- il serializzatore **in sola lettura** per elencare gli oggetti (azione `list`);
- il serializzatore **di creazione** per creare nuovi oggetti (azione `create`) e per generare i metadati del viewset (azione `metadata`);

- il serializzatore ottenuto da `models.SophonGroupModel.get_access_serializer()` per la visualizzazione dettagliata (azione `retrieve`), la modifica (azioni `update` e `partial_update`) e l'eliminazione (azione `destroy`) di un singolo oggetto;
- il serializzatore ottenuto da `get_custom_serializer_classes()` per le azioni personalizzate.

Vedi anche:

`models.SophonGroupModel`

get_custom_serializer_classes(*self*) → t.Type[Serializer]

Permette alle classi che ereditano da questa di selezionare quale `rest_framework.serializers.Serializer` utilizzare per le azioni personalizzate.

class WriteSophonViewSet(*rest_framework.viewsets.ModelViewSet*, *ReadSophonViewSet*, *metaclass=abc.ABCMeta*)

Classe **astratta** che estende la classe base `ReadSophonViewSet` aggiungendoci i metodi di `rest_framework.viewsets.ModelViewSet` che effettuano modifiche sugli oggetti.

Depreca i metodi `perform_*` di `rest_framework`, introducendone versioni estese con una signature diversa dal nome di `hook_*`.

perform_create(*self*, *serializer*)

Deprecato dalla versione 0.1.

Metodo di `rest_framework` rimosso da Sophon.

perform_update(*self*, *serializer*)

Deprecato dalla versione 0.1.

Metodo di `rest_framework` rimosso da Sophon.

perform_destroy(*self*, *serializer*)

Deprecato dalla versione 0.1.

Metodo di `rest_framework` rimosso da Sophon.

hook_create(*self*, *serializer*) → dict[75][str[76], typing.Any]

Funzione chiamata durante l'esecuzione dell'azione di creazione oggetto `create`.

Parametri **serializer** -- Il `Serializer` già "riempito" contenente i dati dell'oggetto che sta per essere creato.

Solleva **HTTPException** -- È possibile interrompere la creazione dell'oggetto con uno specifico codice errore sollevando una `HTTPException` all'interno della funzione.

Ritorna Un `dict`[77] da unire a quello del `Serializer` per formare l'oggetto da creare.

[72] https://docs.python.org/3.8/library/typing.html#typing.Collection
[73] https://docs.python.org/3.8/library/typing.html#typing.Type
[74] https://docs.python.org/3.8/library/typing.html#typing.Type

hook_update(*self*, *serializer*) → dict[78][str[79], t.Any]

Funzione chiamata durante l'esecuzione delle azioni di modifica oggetto `update` e `partial_update`.

Parametri **serializer** -- Il `Serializer` già "riempito" contenente i dati dell'oggetto che sta per essere modificato.

Solleva **HTTPException** -- È possibile interrompere la creazione dell'oggetto con uno specifico codice errore sollevando una `HTTPException` all'interno della funzione.

Ritorna Un `dict`[80] da unire a quello del `Serializer` per formare l'oggetto da modificare.

hook_destroy(*self*, *serializer*) → dict[81][str[82], typing.Any]

Funzione chiamata durante l'esecuzione dell'azione di eliminazione oggetto `destroy`.

Solleva **HTTPException** -- È possibile interrompere la creazione dell'oggetto con uno specifico codice errore sollevando una `HTTPException` all'interno della funzione.

exception sophon.core.errors.**HTTPException**

Tipo di eccezione che è possibile sollevare nei metodi `hook_*` di `WriteSophonViewSet` per interrompere l'azione in corso senza applicare le modifiche.

status: int

Permette di specificare il codice errore con cui rispondere alla richiesta interrotta.

class SophonGroupViewSet(*WriteSophonViewSet*, *metaclass=abc.ABCMeta*)

Classe **astratta** che estende la classe base `WriteSophonViewSet` estendendo gli `hook_*` con verifiche dei permessi dell'utente che tenta di effettuare l'azione.

abstract get_group_from_serializer(*self*, *serializer*) → models.ResearchGroup

Metodo necessario a trovare il gruppo a cui apparterrà un oggetto prima che il suo serializzatore venga elaborato.

Parametri **serializer** -- Il `Serializer` già "riempito" contenente i dati dell'oggetto.

[75] https://docs.python.org/3.8/library/stdtypes.html#dict
[76] https://docs.python.org/3.8/library/stdtypes.html#str
[77] https://docs.python.org/3.8/library/stdtypes.html#dict
[78] https://docs.python.org/3.8/library/stdtypes.html#dict
[79] https://docs.python.org/3.8/library/stdtypes.html#str
[80] https://docs.python.org/3.8/library/stdtypes.html#dict
[81] https://docs.python.org/3.8/library/stdtypes.html#dict
[82] https://docs.python.org/3.8/library/stdtypes.html#str

5.1.2.8 Viewset concreti

Vengono poi definiti tre viewset e una view che permettono interazioni tra l'utente e i modelli definiti nell'app.

`class UsersByIdViewSet`(*ReadSophonViewSet*)
Viewset in sola lettura che permette di recuperare gli utenti dell'istanza partendo dal loro `id`.

Accessibile all'URL `/api/core/users/by-id/ID/`.

`class UsersByUsernameViewSet`(*ReadSophonViewSet*)
Viewset in sola lettura che permette di recuperare gli utenti dell'istanza partendo dal loro `username`.

Accessibile all'URL `/api/core/users/by-username/USERNAME/`.

`class ResearchGroupViewSet`(*WriteSophonViewSet*)
Viewset in lettura e scrittura che permette di interagire con i gruppi di ricerca.

Accessibile all'URL `/api/core/groups/GROUP_SLUG/`.

`join`(*self*, *request: Request*, *pk: int*[83]) → Response
Azione personalizzata che permette ad un utente di unirsi ad un gruppo aperto.

Utilizza `models.SophonGroupModel.get_access_serializer`.

`leave`(*self*, *request: Request*, *pk: int*[84]) → Response
Azione personalizzata che permette ad un utente di abbandonare un gruppo di cui non è proprietario.

Utilizza `models.SophonGroupModel.get_access_serializer`.

`class SophonInstanceDetailsView`(*APIView*)
View che restituisce il valore attuale dell'unico oggetto `models.SophonInstanceDetails`.

Accessibile tramite richieste `GET` all'URL `/api/core/instance/`.

5.1.2.9 Pagina di amministrazione

Vengono infine registrati nella pagina di amministrazione i modelli concreti definiti in questa app, effettuando alcune personalizzazioni elencate in seguito.

`class ResearchGroupAdmin`(*SophonAdmin*)
Per i gruppi di ricerca, viene specificato un ordinamento, permesso il filtraggio e selezionati i campi più importanti da visualizzare nella lista.

`class SophonInstanceDetails`(*SophonAdmin*)
Per i dettagli dell'istanza, vengono disattivate tutte le azioni, impedendo la creazione o eliminazione del singleton.

[83] https://docs.python.org/3.8/library/functions.html#int
[84] https://docs.python.org/3.8/library/functions.html#int

5.1.2.10 Testing in Sophon Core

Per verificare che i *modelli* e *viewset* funzionassero correttamente e non avessero problemi di *sicurezza*, sono stati realizzati degli unit test in grado di rilevare la presenza di errori all'interno dell'app.

5.1.2.11 Test case generici

Vengono definiti alcuni test case generici per facilitare le interazioni tra `APITestCase` e viewset.

Nota: I nomi delle funzioni usano nomi con capitalizzazione inconsistente, in quanto lo stesso modulo `unittest`[85] non rispetta lo stile suggerito in **PEP 8**[86].

`class BetterAPITestCase`(*APITestCase*)

`as_user`(*self*, *username: str*[87], *password: str*[88] = *None*) →
typing.ContextManager[89][None[90]]

Context manager che permette di effettuare richieste all'API come uno specifico utente, effettuando il logout quando sono state effettuate le richieste necessarie.

`assertData`(*self*, *data: ReturnDict*, *expected: dict*[91])

Asserzione che permette di verificare che l'oggetto restituito da una richiesta all'API contenga almeno le chiavi e i valori contenuti nel dizionario `expected`.

`class ReadSophonTestCase`(*BetterAPITestCase*, *metaclass=abc.ABCMeta*)

Classe **astratta** che implementa metodi per testare rapidamente le azioni di un `views.ReadSophonViewSet`.

`classmethod get_basename`(*cls*) → str[92]

Metodo **astratto** che deve restituire il basename del viewset da testare.

`classmethod get_url`(*cls*, *kind: str*[93], **args*, ***kwargs*) → str[94]

Metodo utilizzato dal test case per trovare gli URL ai quali possono essere effettuate le varie azioni.

I seguenti metodi permettono di effettuare azioni sul viewset:

`list`(*self*) → rest_framework.response.Response

`retrieve`(*self*, *pk*) → rest_framework.response.Response

`custom_list`(*self*, *method: str*[95], *action: str*[96], *data: dict*[97] = *None*) →
rest_framework.response.Response

[85] https://docs.python.org/3.8/library/unittest.html#module-unittest
[86] https://www.python.org/dev/peps/pep-0008
[87] https://docs.python.org/3.8/library/stdtypes.html#str
[88] https://docs.python.org/3.8/library/stdtypes.html#str
[89] https://docs.python.org/3.8/library/typing.html#typing.ContextManager
[90] https://docs.python.org/3.8/library/constants.html#None
[91] https://docs.python.org/3.8/library/stdtypes.html#dict

custom_detail(*self*, *method: str*[98], *action: str*[99], *pk*, *data: dict*[100] = *None*) → rest_framework.response.Response

I seguenti metodi asseriscono che una determinata azione con determinati parametri risponderà con il codice di stato `code`, e restituiscono i dati contenuti nella risposta se l'azione è riuscita (`200 ⩽ code < 300`)

assertActionList(*self*, *code: int*[101] = *200*) → typing.Optional[ReturnDict]

assertActionRetrieve(*self*, *pk*, *code: int*[102] = *200*) → typing.Optional[ReturnDict]

assertActionCustomList(*self*, *method: str*[103], *action: str*[104], *data: dict*[105] = *None*, *code: int*[106] = *200*) → typing.Optional[ReturnDict]

assertActionCustomDetail(*self*, *method: str*[107], *action: str*[108], *pk*, *data: dict*[109] = *None*, *code: int*[110] = *200*) → typing.Optional[ReturnDict]

class WriteSophonTestCase(*ReadSophonTestCase*, *metaclass=abc.ABCMeta*)

Classe **astratta** che estende `ReadSophonTestCase` con le azioni di un `views.WriteSophonViewSet`.

create(*self*, *data*) → rest_framework.response.Response

update(*self*, *pk*, *data*) → rest_framework.response.Response

destroy(*self*, *pk*) → rest_framework.response.Response

assertActionCreate(*self*, *data*, *code: int*[111] = *201*) → typing.Optional[ReturnDict]

assertActionUpdate(*self*, *pk*, *data*, *code: int*[112] = *200*) → typing.Optional[ReturnDict]

assertActionDestroy(*self*, *pk*, *code: int*[113] = *200*) → typing.Optional[ReturnDict]

[92] https://docs.python.org/3.8/library/stdtypes.html#str
[93] https://docs.python.org/3.8/library/stdtypes.html#str
[94] https://docs.python.org/3.8/library/stdtypes.html#str
[95] https://docs.python.org/3.8/library/stdtypes.html#str
[96] https://docs.python.org/3.8/library/stdtypes.html#str
[97] https://docs.python.org/3.8/library/stdtypes.html#dict
[98] https://docs.python.org/3.8/library/stdtypes.html#str
[99] https://docs.python.org/3.8/library/stdtypes.html#str
[100] https://docs.python.org/3.8/library/stdtypes.html#dict
[101] https://docs.python.org/3.8/library/functions.html#int
[102] https://docs.python.org/3.8/library/functions.html#int
[103] https://docs.python.org/3.8/library/stdtypes.html#str
[104] https://docs.python.org/3.8/library/stdtypes.html#str
[105] https://docs.python.org/3.8/library/stdtypes.html#dict
[106] https://docs.python.org/3.8/library/functions.html#int
[107] https://docs.python.org/3.8/library/stdtypes.html#str
[108] https://docs.python.org/3.8/library/stdtypes.html#str
[109] https://docs.python.org/3.8/library/stdtypes.html#dict
[110] https://docs.python.org/3.8/library/functions.html#int

5.1.2.12 Test case concreti

Vengono testate tutte le view dell'app tramite `BetterAPITestCase` e tutti i viewset dell'app tramite `ReadSophonTestCase` e `WriteSophonTestCase`.

`class UsersByIdTestCase`(*ReadSophonTestCase*)

`class UsersByUsernameTestCase`(*ReadSophonTestCase*)

`class ResearchGroupTestCase`(*WriteSophonTestCase*)

`class SophonInstanceDetailsTestCase`(*BetterAPITestCase*)

5.1.3 L'app Sophon Projects

L'app `sophon.projects` è un app secondaria che dipende da `sophon.core` che introduce in Sophon il concetto di *progetto di ricerca*.

Nota: L'app `sophon.projects` teoricamente è opzionale, in quanto il modulo backend può funzionare senza di essa, e può essere rimossa dal modulo `sophon.settings`.

Non è però possibile rimuoverla nella versione finale distribuita, in quanto il modulo `sophon.settings` non è modificabile dall'esterno, e in quanto il *modulo frontend* non prevede questa funzionalità e si aspetta che i percorsi API relativi all'app siano disponibili.

Inoltre, rimuovendo l'app `sophon.projects` non sarà più possibile usare l'app `sophon.notebooks`, in quanto dipende da essa.

5.1.3.1 Modello del progetto di ricerca

Viene introdotto un modello concreto che rappresenta un *progetto di ricerca*.

`class ResearchProject`(*SophonGroupModel*)

`slug: SlugField`
: L'identificatore del progetto di ricerca, usato nei percorsi dell'API.

`group: ForeignKey → sophon.core.models.ResearchGroup`
: Lo `slug` del gruppo di ricerca al quale appartiene il progetto.

`name: CharField`
: Il nome completo del progetto di ricerca.

`description: TextField`
: La descrizione del progetto di ricerca, da visualizzare in un riquadro "A proposito del progetto".

[111] https://docs.python.org/3.8/library/functions.html#int
[112] https://docs.python.org/3.8/library/functions.html#int
[113] https://docs.python.org/3.8/library/functions.html#int

`visibility: CharField ["PUBLIC", "INTERNAL", "PRIVATE"]`
La *visibilità del progetto*.

5.1.3.2 Viewset del gruppo di ricerca

Da una base comune, vengono creati due viewset per interagire con i progetti di ricerca.

`class ResearchProjectViewSet`(*SophonGroupViewSet*, *metaclass=abc.ABCMeta*)
Classe **astratta** che effettua l'override di `get_group_from_serializer()` per entrambi i viewset che seguono.

`class ResearchProjectsBySlugViewSet`(*ResearchProjectViewSet*)
Viewset in lettura e scrittura che permette di interagire con tutti i progetti di ricerca a cui l'utente loggato ha accesso.

Accessibile all'URL `/api/projects/by-slug/PROJECT_SLUG/`.

`class ResearchProjectsByGroupViewSet`(*ResearchProjectViewSet*)
Viewset in lettura e scrittura che permette di interagire con i progetti di ricerca a cui l'utente loggato ha accesso, filtrati per il gruppo a cui appartengono.

Il filtraggio viene effettuato limitando il queryset.

Accessibile all'URL `/api/projects/by-group/GROUP_SLUG/PROJECT_SLUG/`.

5.1.3.3 Amministrazione del gruppo di ricerca

Il modello `models.ResearchProject` viene registrato nella pagina di amministrazione attraverso la seguente classe:

`class ResearchProjectAdmin`(*sophon.core.admin.SophonAdmin*)
Classe per la pagina di amministrazione che specifica un ordinamento, permette il filtraggio per gruppo di appartenenza e visibilità, e specifica i campi da visualizzare nell'elenco dei progetti.

5.1.4 L'app Sophon Notebooks

L'app `sophon.notebooks` è un app secondaria che dipende da `sophon.projects` che introduce in Sophon il concetto di `notebook`.

Nota: L'app `sophon.notebooks` teoricamente è opzionale, in quanto il modulo backend può funzionare senza di essa, e può essere rimossa dal modulo `sophon.settings`.

Non è però possibile rimuoverla nella versione finale distribuita, in quanto il modulo `sophon.settings` non è modificabile dall'esterno, e in quanto il `modulo frontend` non prevede questa funzionalità e si aspetta che i percorsi API relativi all'app siano disponibili.

5.1.4.1 Funzionamento di un notebook

Internamente, un notebook non è altro che un container *Docker* accessibile ad un determinato indirizzo il cui stato è sincronizzato con un oggetto del database del `modulo backend`.

5.1.4.1.1 Modalità sviluppo

Per facilitare lo sviluppo di Sophon, sono state realizzate due modalità di operazione di quest'ultimo.

- Nella prima, la **modalità sviluppo**, il `modulo proxy` non è in esecuzione, ed è possibile collegarsi direttamente ai container all'indirizzo IP locale `127.0.0.1`.

  Il `modulo frontend` non supporta questa modalità, in quanto intesa solamente per lo sviluppo del modulo backend.

- Nella seconda, la **modalità produzione**, il `modulo proxy` è in esecuzione all'interno di un container Docker, e si collega ai `moduli Jupyter` attraverso i relativi network Docker tramite una `rubrica`.

5.1.4.2 Gestione della rubrica del proxy

Viene creata una classe per la gestione della rubrica del proxy, utilizzando il modulo `dbm.gnu`[114], supportato da HTTPd.

La rubrica mappa gli URL pubblici dei notebook a URL privati relativi al `modulo proxy`, in modo da effettuare reverse proxying **dinamico**.

`class ApacheDB`

Classe che permette il recupero, la creazione, la modifica e l'eliminazioni di chiavi di un database `dbm.gnu`[115] come se quest'ultimo fosse un `dict`[116] con supporto a chiavi e valori `str`[117] e `bytes`[118].

`static convert_to_bytes`(*item: typing.Union[str*[119]*, bytes*[120]*]*) → bytes[121]

Tutte le `str`[122] passate a questa classe vengono convertite in `bytes`[123] attraverso questa funzione, che effettua un encoding in ASCII e solleva un errore se quest'ultimo fallisce.

[114] https://docs.python.org/3.8/library/dbm.html#module-dbm.gnu
[115] https://docs.python.org/3.8/library/dbm.html#module-dbm.gnu
[116] https://docs.python.org/3.8/library/stdtypes.html#dict
[117] https://docs.python.org/3.8/library/stdtypes.html#str
[118] https://docs.python.org/3.8/library/stdtypes.html#bytes
[119] https://docs.python.org/3.8/library/stdtypes.html#str
[120] https://docs.python.org/3.8/library/stdtypes.html#bytes
[121] https://docs.python.org/3.8/library/stdtypes.html#bytes
[122] https://docs.python.org/3.8/library/stdtypes.html#str
[123] https://docs.python.org/3.8/library/stdtypes.html#bytes

5.1.4.3 Assegnazione porta effimera

In *modalità sviluppo*, è necessario trovare una porta libera a cui rendere accessibile i container Docker dei notebook.

`get_ephemeral_port``()` → int[124]

Questa funzione apre e chiude immediatamente un `socket.socket`[125] all'indirizzo `localhost:0` in modo da ricevere dal sistema operativo un numero di porta sicuramente libero.

5.1.4.4 Connessione al daemon Docker

Per facilitare l'utilizzo del daemon Docker per la gestione dei container dei notebook, viene utilizzato il modulo `docker`.

`get_docker_client``()` → docker.DockerClient

Funzione che crea un client Docker con le variabili di ambiente del modulo.

`client: docker.DockerClient = lazy_object_proxy.Proxy(get_docker_client)`

Viene creato un client Docker globale con inizializzazione lazy al fine di non tentare connessioni (lente!) al daemon quando non sono necessarie.

5.1.4.5 Controllo dello stato di salute

Il modulo `docker` viene esteso implementando supporto per l'istruzione `HEALTHCHECK` dei `Dockerfile`.

`class HealthState``(`*enum.IntEnum*`)`

Enumerazione che elenca gli stati possibili in cui può essere la salute di un container.

`UNDEFINED = -2`

Il `Dockerfile` non ha un `HEALTHCHECK` definito.

`STARTING = -1`

Il container Docker non mai completato con successo un `HEALTHCHECK`.

`HEALTHY = 0`

Il container Docker ha completato con successo l'ultimo `HEALTHCHECK` e quindi sta funzionando correttamente.

`UNHEALTHY = 1`

Il container Docker ha fallito l'ultimo `HEALTHCHECK`.

`get_health``(`*container: docker.models.containers.Container*[126]`)` → *HealthState*

Funzione che utilizza l'API a basso livello del client Docker per recuperare l'`HealthState` dei container.

[124] https://docs.python.org/3.8/library/functions.html#int
[125] https://docs.python.org/3.8/library/socket.html#socket.socket
[126] https://docker-py.readthedocs.io/en/stable/containers.html#docker.models.containers.Container

sleep_until_container_has_started(*container: docker.models.containers.Container*[127]) → *HealthState*

Funzione bloccante che restituisce solo quando lo stato del container specificato non è `HealthState.STARTING`.

> **Pericolo:** L'implementazione di questa funzione potrebbe causare rallentamenti nella risposta alle pagine web per via di una chiamata al metodo `time.sleep`[128] al suo interno.
>
> Ciò è dovuto al mancato supporto alle funzioni asincrone nella versione attuale di `rest_framework`.
>
> Si è deciso di mantenere comunque la funzionalità a scopi dimostrativi e per compatibilità futura.

5.1.4.6 Generazione di token sicuri

Per rendere l'interfaccia grafica più `intuitiva`, si è scelto di rendere trasparente all'utente il meccanismo di autenticazione a JupyterLab.

Pertanto, si è verificata la necessità di generare token crittograficamente sicuri da richiedere per l'accesso a JupyterLab.

generate_secure_token() → str[129]

Funzione che utilizza `secrets.token_urlsafe`[130] per generare un token valido e crittograficamente sicuro.

5.1.4.7 Modello dei notebook

Viene definito il modello rappresentante un `notebook`.

class Notebook(*SophonGroupModel*)

slug: SlugField

Lo slug dei notebook prevede ulteriori restrizioni oltre a quelle previste dallo `django.db.models.SlugField`[131]:

- non può essere uno dei seguenti valori: `api`, `static`, `proxy`, `backend`, `frontend`, `src`;
- non può iniziare o finire con un trattino `-`.

[127] https://docker-py.readthedocs.io/en/stable/containers.html#docker.models.containers.Container
[128] https://docs.python.org/3.8/library/time.html#time.sleep
[129] https://docs.python.org/3.8/library/stdtypes.html#str
[130] https://docs.python.org/3.8/library/secrets.html#secrets.token_urlsafe

project: ForeignKey → sophon.projects.models.ResearchProject
Il `progetto` che include questo notebook.

name: CharField
Il nome del notebook.

locked_by: ForeignKey → django.contrib.auth.models.User
L'`utente` che ha richiesto il blocco del notebook, o `None`[132] in caso il notebook non sia bloccato.

container_image: CharField ["ghcr.io/steffo99/sophon-jupyter"]
Campo che specifica l'immagine che il client *Docker* dovrà avviare per questo notebook.

Al momento ne è supportata una sola per semplificare l'esperienza utente, ma altre possono essere aggiunte al file che definisce il modello per permettere agli utenti di scegliere tra più immagini.

Nota: Al momento, Sophon si aspetta che tutte le immagini specificate espongano un server web sulla porta `8888`, e supportino il protocollo di autenticazione di Jupyter, ovvero che sia possibile raggiungere il container ai seguenti indirizzi: `PROTOCOLLO://immagine:8888/lab?token=TOKEN` e `PROTOCOLLO://immagine:8888/tree?token=TOKEN`.

---

jupyter_token: CharField
Il token segreto che verrà passato attraverso le variabili di ambiente al container Docker dell'oggetto per permettere solo agli utenti autorizzati di accedere a quest'ultimo.

container_id: CharField
L'id assegnato dal daemon Docker al container di questo oggetto.

Se il notebook non è avviato, questo attributo varrà `None`[133].

port: IntegerField
La porta TCP locale assegnata al container Docker dell'oggetto nel caso in cui Sophon sia avviato in `modalità sviluppo`.

internal_url: CharField
L'URL a cui è accessibile il container Docker dell'oggetto nel caso in cui Sophon non sia avviato in `modalità sviluppo`.

property log(*self*) → logging.Logger[134]
Viene creato un `logging.Logger`[135] per ogni oggetto della classe, in modo da facilitare il debug relativo ad uno specifico notebook.

Il nome del logger ha la forma `sophon.notebooks.models.Notebook.NOTEBOOK_SLUG`.

enable_proxying(*self*) → None[136]
Aggiunge l'indirizzo del notebook alla `rubrica del proxy`.

disable_proxying(*self*) → None[137]
Rimuove l'indirizzo del notebook dalla `rubrica del proxy`.

sync_container(*self*) → t.Optional[docker.models.containers.Container[138]]
Sincronizza lo stato dell'oggetto nel database con lo stato del container *Docker* nel sistema.

create_container(*self*) → docker.models.containers.Container[139]
Crea e configura un container *Docker* per l'oggetto, con l'immagine specificata in `container_image`.

start(*self*) → None[140]
Tenta di creare e avviare un container *Docker* per l'oggetto, bloccando fino a quando esso non sarà avviato con `sleep_until_container_has_started`.

stop(*self*) → None[141]
Arresta il container Docker dell'oggetto.

5.1.4.8 Viewset dei notebook

Come per il modulo `sophon.projects`, vengono creati due viewset per interagire con i progetti di ricerca, basati entrambi su un viewset astratto che ne definisce le proprietà comuni.

class NotebooksViewSet(*SophonGroupViewSet*, *metaclass=abc.ABCMeta*)
Classe **astratta** che effettua l'override di `get_group_from_serializer` e definisce cinque azioni personalizzate per l'interazione con il notebook.

sync(*self*, *request: Request*, ***kwargs*) → Response
Azione personalizzata che sincronizza lo stato dell'oggetto dell'API con quello del daemon Docker.

start(*self*, *request: Request*, ***kwargs*) → Response
Azione personalizzata che avvia il notebook con `models.Notebook.start`.

stop(*self*, *request: Request*, ***kwargs*) → Response
Azione personalizzata che arresta il notebook con `models.Notebook.stop`.

lock(*self*, *request: Request*, ***kwargs*) → Response
Azione personalizzata che blocca il notebook impostando il campo `models.Notebook.locked_by` all'utente che ha effettuato la richiesta.

[131] http://docs.djangoproject.com/en/3.2/ref/models/fields/#django.db.models.SlugField
[132] https://docs.python.org/3.8/library/constants.html#None
[133] https://docs.python.org/3.8/library/constants.html#None
[134] https://docs.python.org/3.8/library/logging.html#logging.Logger
[135] https://docs.python.org/3.8/library/logging.html#logging.Logger
[136] https://docs.python.org/3.8/library/constants.html#None
[137] https://docs.python.org/3.8/library/constants.html#None
[138] https://docker-py.readthedocs.io/en/stable/containers.html#docker.models.containers.Container
[139] https://docker-py.readthedocs.io/en/stable/containers.html#docker.models.containers.Container
[140] https://docs.python.org/3.8/library/constants.html#None
[141] https://docs.python.org/3.8/library/constants.html#None

unlock(*self*, *request: Request*, ***kwargs*) → Response

Azione personalizzata che sblocca il notebook impostando il campo `models.Notebook.locked_by` a `None`[142].

class NotebooksBySlugViewSet(*NotebooksViewSet*)

Viewset in lettura e scrittura che permette di interagire con tutti i notebook a cui l'utente loggato ha accesso.

Accessibile all'URL `/api/notebooks/by-slug/NOTEBOOK_SLUG/`.

class NotebooksByProjectViewSet(*NotebooksViewSet*)

Viewset in lettura e scrittura che permette di interagire con i notebook a cui l'utente loggato ha accesso, filtrati per il progetto di appartenenza.

Accessibile all'URL `/api/notebooks/by-project/PROJECT_SLUG/NOTEBOOK_SLUG/`.

5.1.5 Containerizzazione del modulo backend

Il modulo backend è incapsulato in un'immagine *Docker* basata sull'immagine ufficiale python:3.9.7-bullseye[143].

L'immagine utilizza `Poetry` per installare le dipendenze, poi esegue il file `docker_start.sh` riportato sotto che effettua le migrazioni, prepara i file statici di Django e `prova a creare un superutente`, per poi avviare il progetto Django attraverso `gunicorn` sulla porta 8000.

```
poetry run python -O ./manage.py migrate --no-input
poetry run python -O ./manage.py collectstatic --no-input
poetry run python -O ./manage.py initsuperuser
poetry run python -O -m gunicorn sophon.wsgi:application -
↪-workers=4 --bind=0.0.0.0:8000 --timeout 180
```

5.2 Realizzazione del modulo frontend

Il modulo frontend è stato realizzato come un package `Node.js` denominato `@steffo/sophon-frontend`, e poi `containerizzato`, creando un'immagine *Docker* standalone, esattamente come per il `modulo backend`.

[142] https://docs.python.org/3.8/library/constants.html#None
[143] https://hub.docker.com/_/python

5.2.1 Struttura delle directory

Le directory di `@steffo45/sophon-frontend` sono strutturate nella seguente maniera:

src/components Contiene i componenti React sia con le classi sia funzionali.

src/contexts Contiene i contesti React creati con `React.createContext()`.

src/hooks Contiene gli hook React personalizzati utilizzati nei componenti funzionali.

src/types Contiene estensioni ai tipi base TypeScript, come ad esempio i tipi restituiti dalla web API del *Modulo backend*.

src/utils Contiene varie funzioni di utility.

public Contiene i file statici da servire assieme all'app.

5.2.2 Comunicazione con il backend

Sono state sviluppate alcune funzioni di utilità per facilitare la comunicazione con il `modulo backend`.

5.2.2.1 Axios

Per effettuare richieste all'API web, si è deciso di utilizzare la libreria `axios`, in quanto permette di creare dei "client" personalizzabili con varie proprietà.

In particolare, si è scelto di effettuare un fork della stessa, integrando anticipatamente una proposta di funzionalità che permette alle richieste di essere interrotte attraverso degli `AbortController()`.

5.2.2.2 Client personalizzati

Per permettere all'utente di selezionare l'istanza da utilizzare e di comunicare con l'API con le proprie credenziali, si è scelto di creare client personalizzati partendo da due contesti.

All'interno di un contesto in cui è stata selezionata un'istanza (`InstanceContext`), viene creato un client dal seguente hook:

`useInstanceAxios`(*config = {}*)
: Questo hook specifica il `baseURL` del client Axios, impostandolo all'URL dell'istanza selezionata.

All'interno di un contesto in cui è stato effettuato l'accesso come utente (`AuthorizationContext`), viene creato invece un client dal seguente hook:

`useAuthorizedAxios`(*config = {}*)
: Questo hook specifica il valore dell'header `Authorization` da inviare in tutte le richieste effettuate a `Bearer TOKEN`, utilizzando il token ottenuto al momento dell'accesso.

5.2.2.3 Utilizzo di viewset

Viene implementato un hook che si integra con i viewset di Django, fornendo un API semplificato per effettuare azioni su di essi.

`useViewSet(baseRoute) → viewset()`

Questo hook implementa tutte le azioni `rest_framework` di un viewset in lettura e scrittura.

Richiede di essere chiamato all'interno di un `AuthorizationContext`.

`viewset.`**`list`**`(`*config = {}*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Richiede la lista di tutte le risorse del viewset.

`viewset.`**`retrieve`**`(`*pk*, *config = {}*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Richiede i dettagli di una specifica risorsa del viewset.

`viewset.`**`create`**`(`*config*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Crea una nuova risorsa nel viewset.

`viewset.`**`update`**`(`*pk*, *config*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Aggiorna una specifica risorsa nel viewset.

`viewset.`**`destroy`**`(`*pk*, *config*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Elimina una specifica risorsa dal viewset.

Viene inoltre fornito supporto per le azioni personalizzate.

`viewset.`**`command`**`(`*config*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Permette azioni personalizzate su tutto il viewset.

`viewset.`**`action`**`(`*config*`)`

Funzione **asincrona**, che restituisce una `Promise()`.

Permette azioni personalizzate su uno specifico oggetto del viewset.

5.2.2.4 Emulazione di viewset

Viene creato un hook che tiene traccia degli oggetti restituiti da un determinato viewset, ed emula i risultati delle azioni effettuate, minimizzando i rerender e ottenendo una ottima user experience.

`useManagedViewSet(baseRoute, pkKey, refreshOnMount) → managed()`

`managed.`**`viewset`**
: Il viewset restituito da `useViewSet()`, utilizzato come interfaccia di basso livello per effettuare azioni.

`managed.`**`state`**
: Lo stato del viewset, che tiene traccia degli oggetti e delle azioni in corso su di essi.

  Gli oggetti all'interno di esso sono istanze di `ManagedResource()`, create usando wrapper di `update()`, `destroy()` e `action()`, che permettono di modificare direttamente l'oggetto senza preoccuparsi dell'indice a cui si trova nell'array.

`managed.`**`dispatch`**
: Riduttore che permette di alterare lo `state`.

`managed.`**`refresh`**`()`
: Funzione **asincrona**, che restituisce una `Promise()`.

  Ricarica gli oggetti del viewset.

  Viene chiamata automaticamente al primo render se `refreshOnMount` è `True`.

`managed.`**`create`**`(`*data*`)`
: Funzione **asincrona**, che restituisce una `Promise()`.

  Crea un nuovo oggetto nel viewset con i dati specificati come argomento, e lo aggiunge allo stato se la richiesta va a buon fine.

`managed.`**`command`**`(`*method*, *cmd*, *data*`)`
: Funzione **asincrona**, che restituisce una `Promise()`.

  Esegue l'azione personalizzata `cmd` su tutto il viewset, utilizzando il metodo `method` e con i dati specificati in `data`.

  Se la richiesta va a buon fine, il valore restituito dal backend sostituisce nello stato le risorse dell'intero viewset.

`managed.`**`update`**`(`*index*, *data*`)`
: Funzione **asincrona**, che restituisce una `Promise()`.

  Modifica l'oggetto alla posizione `index` dell'array `state` con i dati specificati in `data`.

  Se la richiesta va a buon fine, la modifica viene anche applicata all'interno di `state`

`managed.`**`destroy`**`(`*index*`)`
: Funzione **asincrona**, che restituisce una `Promise()`.

  Elimina l'oggetto alla posizione `index` dell'array `state`.

Se la richiesta va a buon fine, l'oggetto viene eliminato anche da `state`.

`managed.action(`*index*, *method*, *act*, *data*`)`
Funzione **asincrona**, che restituisce una `Promise()`.

Esegue l'azione personalizzata `act` sull'oggetto alla posizione `index` dell'array `state`, utilizzando il metodo `method` e con i dati specificati in `data`.

Se la richiesta va a buon fine, il valore restituito dal backend sostituisce l'oggetto utilizzato in `state`.

5.2.3 Contesti innestati

Per minimizzare i re-render, l'applicazione è organizzata a "contesti innestati".

5.2.3.1 I contesti

Viene definito un contesto per ogni tipo di risorsa selezionabile nell'interfaccia.

Essi sono, in ordine dal più esterno al più interno:

1. `InstanceContext` (`Istanza`)
2. `AuthorizationContext` (`Utente`)
3. `GroupContext` (`Gruppo di ricerca`)
4. `ProjectContext` (`Progetto di ricerca`)
5. `NotebookContext` (`Notebook`)

5.2.3.1.1 Contenuto dei contesti

Questi contesti possono avere tre tipi di valori: `undefined` se ci si trova al di fuori del contesto, `null` se non è stato selezionato alcun oggetto oppure **l'oggetto selezionato** se esso esiste.

5.2.3.2 Segmenti di URL

Si è definita la seguente struttura per gli URL del frontend di Sophon, in modo che essi identificassero universalmente una risorsa e che essi fossero human-readable.

```
/i/{ISTANZA}
   /l/logged-in
      /g/{GROUP_SLUG}
         /p/{PROJECT_SLUG}
            /n/{NOTEBOOK_SLUG}
               /
```

Ciascuna riga nel listato sopra rappresenta un *segmento di URL*.

5.2.3.2.1 Parsing dei segmenti di URL

Il parsing di questi segmenti viene effettuato dalla seguente funzione:

`parsePathSegment({path, parsed, regex, key, next}) → ParsedPath()`

Funzione ricorsiva per la cattura di un segmento, che riempie ad ogni chiamata una chiave dell'oggetto `ParsedPath`.

Parametri

- **`path`** -- La stringa del percorso ancora da parsare.
- **`parsed`** -- L'oggetto `ParsedPath` riempito con i segmenti trovati fino ad ora.
- **`regex`** -- Una regular expression usata per catturare un segmento. Il **primo gruppo di cattura** sarà il valore che verrà mantenuto e inserito nel `ParsedPath`.
- **`key`** -- La chiave a cui inserire il valore catturato all'interno del `ParsedPath`.
- **`next`** -- Callback della prossima funzione da chiamare.

Un esempio di URL per il notebook `my-first-notebook` all'interno della istanza demo di Sophon potrebbe essere:

```
/i/https:api.prod.sophon.steffo.eu:
   /l/logged-in
      /g/my-first-group
         /p/my-first-project
            /n/my-first-notebook
               /
```

5.2.3.2.2 Breadcrumbs

È possibile vedere tutti i segmenti di URL dalla barra superiore dell'interfaccia grafica, detta *barra dei breadcrumbs*.

Figura 5.2.1: La barra dei breadcrumbs. Ci si trova attualmente sulla pagina del gruppo `my-first-group`.

Nel caso il parsing dei segmenti fallisca, la barra dei breadcrumbs è in grado di mostrare un errore, e di permettere all'utente di riprendere la navigazione ad uno dei segmenti trovati.

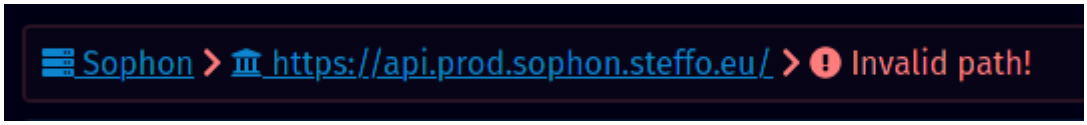


Figura 5.2.2: La barra dei breadcrumbs in errore. È possibile riprendere la navigazione dalla pagina di selezione istanza o di login.

5.2.3.3 Componenti contestuali

Per ciascun contesto sono stati realizzati vari componenti.

I più significativi comuni a tutti i contesti sono i `ResourcePanel` e le `ListBox`.

`ResourcePanel``(`*`{...}`*`)`
: Panello che rappresenta un'`entità di Sophon`, diviso in quattro parti:

  - icona (a sinistra)
  - nome della risorsa (a destra dell'icona)
  - bottoni (a destra)
  - testo (a sinistra dei bottoni)

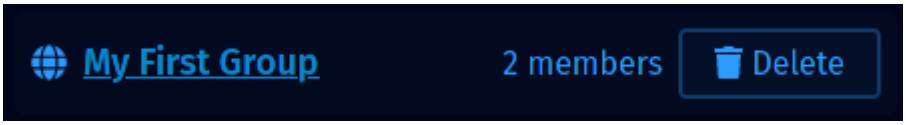


Figura 5.2.3: Un `ResourcePanel` rappresentante un `gruppo di ricerca`.

`ListBox``(`*`{...}`*`)`
: Riquadro che mostra le risorse di un `useManagedViewSet` raffigurandole come tanti `ResourcePanel`.

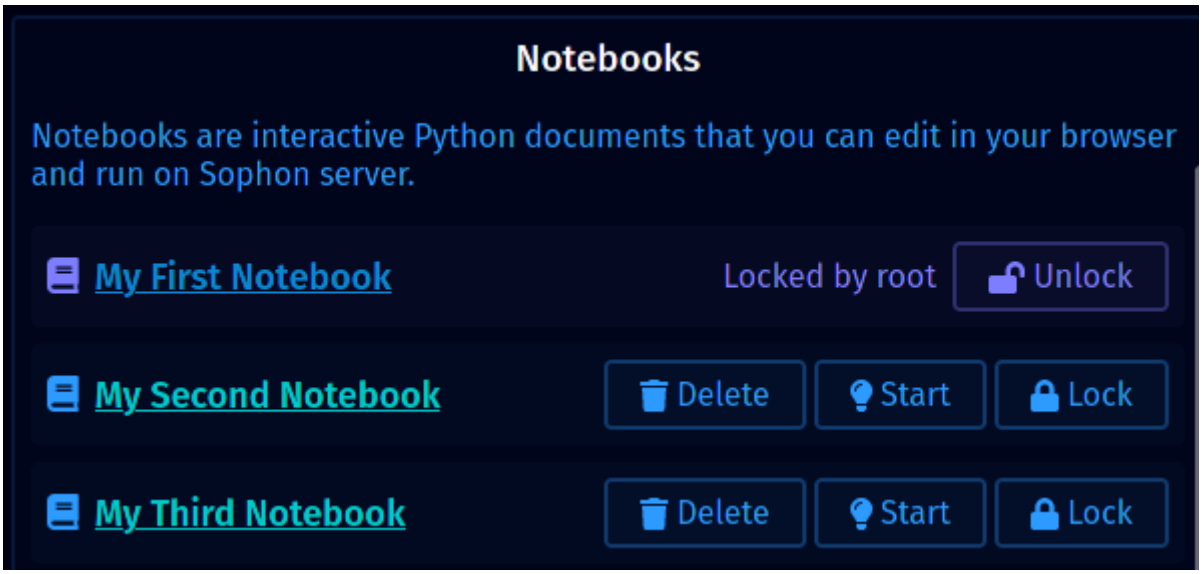


Figura 5.2.4: Una `ListBox` che mostra l'elenco di notebook in un progetto.

5.2.3.4 Routing basato sui contesti

I valori dei contesti vengono utilizzati per selezionare i componenti da mostrare all'utente nell'interfaccia grafica attraverso i seguenti componenti:

`ResourceRouter`(*{selection, unselectedRoute, selectedRoute}*)
: Componente che sceglie se renderizzare `unselectedRoute` o `selectedRoute` in base alla *nullità* o *non-nullità* di `selection`.

`ViewSetRouter`(*{viewSet, unselectedRoute, selectedRoute, pathSegment, pkKey}*)
: Componente basato su `ResourceRouter()` che seleziona automaticamente l'elemento del viewset avente il valore del segmento di percorso `pathSegment` alla chiave `pkKey`.

Ad esempio, `ViewSetRouter()` viene esteso specificatamente per il contesto del gruppo, creando il seguente componente.

`GroupRouter`(*{...props}*)
: Implementato come:

```
<ViewSetRouter
    {...props}
    viewSet={useManagedViewSet<SophonResearchGroup>("/api/
↪core/groups/", "slug")}
    pathSegment={"researchGroup"}
    pkKey={"slug"}
/>
```

5.2.3.5 Albero completo dei contesti

L'insieme di tutti i contesti è definito come componente `App()` nel modulo "principale" `App.tsx`.

Se ne riassume la struttura in pseudocodice:

```
<InstanceContext>
   <InstanceRouter>
      unselected:
         <InstanceSelect>
      selected:
         <AuthorizationContext>
            <AuthorizationRouter>
               unselected:
                  <UserLogin>
               selected:
                  <GroupContext>
                     <GroupRouter>
                        unselected:
                           <GroupSelect>
                        selected:
                           <ProjectContext>
                              <ProjectRouter>
```

(continues on next page)

(continua dalla pagina precedente)

```
                                    unselected:
                                       <ProjectSelect>
                                    selected:
                                       <NotebookContext>
                                          <NotebookRouter>
                                             unselected:
                                                <NotebookSelect>
                                             selected:
                                                <NotebookDetails>
```

5.2.3.6 Altri contesti

All'interno di Sophon sono presenti anche altri due contesti, utilizzati a scopo di semplificare e ottimizzare il codice.

5.2.3.6.1 Tema

Il tema dell'istanza è implementato come uno speciale contesto globale `ThemeContext` che riceve i dettagli dell'istanza a cui si è collegati dall'`InstanceContext`.

Ciò permette di sincronizzare il tema della webapp con quello dell'istanza di Sophon selezionata.

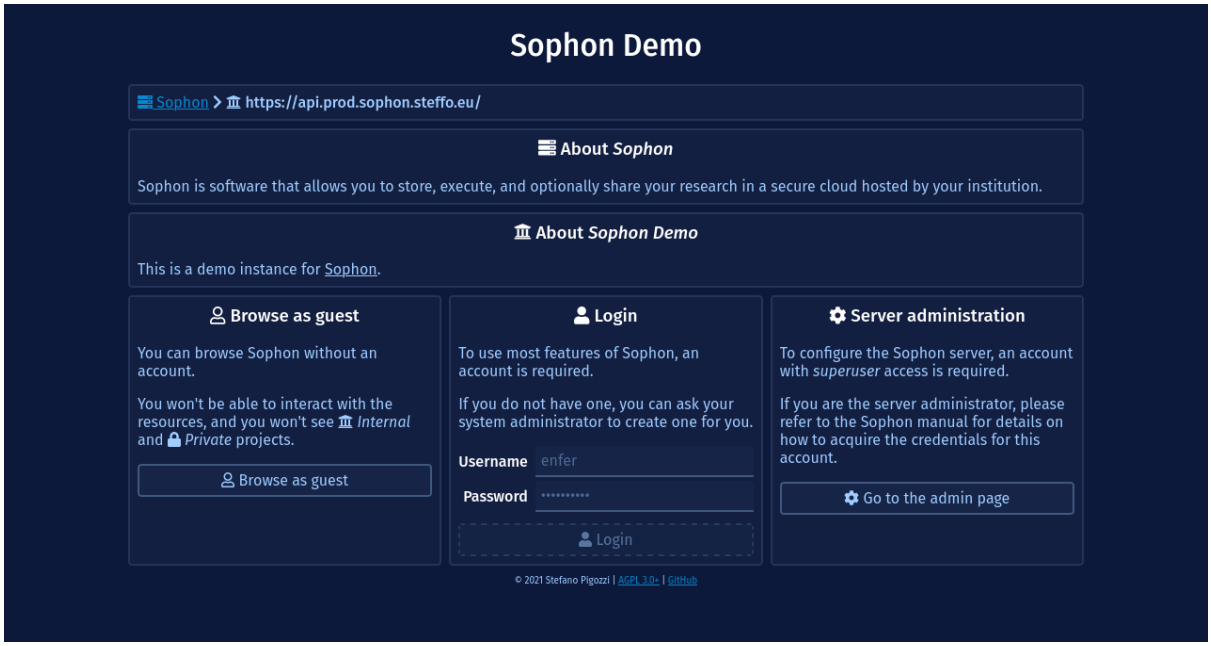


Figura 5.2.5: La schermata di login dell'istanza dimostrativa di Sophon, che utilizza il tema "Royal Blue".

5.2.3.6.2 Cache

Viene salvato l'elenco di tutti i membri dell'`istanza` in uno speciale contesto `CacheContext` in modo da poter risolvere gli id degli utenti al loro username senza dover effettuare ulteriori richieste.

Questa funzionalità al momento viene usata per risolvere i nomi dei membri in un gruppo e il nome dell'utente che ha bloccato un notebook: in entrambi i casi, vengono restituiti dal `modulo backend` solamente gli ID numerici dei relativi utenti, pertanto è necessario risolverli attraverso il contesto cache.

Figura 5.2.6: L'elenco dei membri appartenenti al gruppo "My First Group".

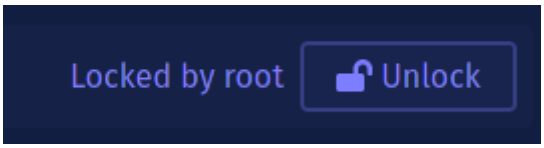


Figura 5.2.7: Il nome di un utente che ha bloccato un notebook. (In questo caso, "root".)

5.2.4 Containerizzazione del modulo frontend

Il modulo frontend è incapsulato in un'immagine *Docker* basata sull'immagine ufficiale node:16.11.1-bullseye[144].

L'immagine installa le dipendenze del modulo con `Yarn`, per poi eseguire il comando `yarn run serve`, che avvia la procedura di preparazione della pagina e la rende disponibile su un webserver locale alla porta 3000.

5.3 Realizzazione del modulo proxy

Il modulo proxy consiste in un file di configurazione di `Apache HTTP Server`.

Il file di configurazione abilita i moduli httpd rewrite[145], proxy[146], proxy_wstunnel[147] e proxy_http[148], impostando quest'ultimo per inoltrare l'header Host[149] alle pagine verso cui viene effettuato reverse proxying.

[144] https://hub.docker.com/_/node
[145] https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html
[146] https://httpd.apache.org/docs/2.4/mod/mod_proxy.html
[147] https://httpd.apache.org/docs/2.4/mod/mod_proxy_wstunnel.html
[148] https://httpd.apache.org/docs/2.4/mod/mod_proxy_http.html
[149] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Host

Inoltre, nel file di configurazione viene abilitato il `RewriteEngine`, che viene utilizzato per effettuare reverse proxying secondo le seguenti regole:

1. Tutte le richieste verso `static.` prefisso ad `APACHE_PROXY_BASE_DOMAIN` vengono processate direttamente dal webserver, utilizzando i file disponibili nella cartella `/var/www/html/django-static` che gli vengono forniti dal volume `django-static` del *Modulo backend*.

```
# If ENV:APACHE_PROXY_BASE_DOMAIN equals HTTP_HOST
RewriteCond "static.%{ENV:APACHE_PROXY_BASE_DOMAIN} %{HTTP_
 ↪HOST}" "^([^ ]+) \1$" [NC]
# Process the request yourself
RewriteRule ".?" - [L]
```

2. Tutte le richieste verso `APACHE_PROXY_BASE_DOMAIN` senza nessun sottodominio vengono inoltrate al container Docker del *Modulo frontend* utilizzando la risoluzione dei nomi di dominio di Docker Compose.

```
# If ENV:APACHE_PROXY_BASE_DOMAIN equals HTTP_HOST
RewriteCond "%{ENV:APACHE_PROXY_BASE_DOMAIN} %{HTTP_HOST}" "^
 ↪([^ ]+) \1$" [NC]
# Capture ENV:SOPHON_FRONTEND_NAME for substitution in the␣
 ↪rewriterule
RewriteCond "%{ENV:SOPHON_FRONTEND_NAME}" "^(.+)$" [NC]
# Forward to the frontend
RewriteRule "/(.*)" "http://%1/$1" [P,L]
```

3. Tutte le richieste verso `api.` prefisso ad `APACHE_PROXY_BASE_DOMAIN` vengono inoltrate al container Docker del *Modulo backend* utilizzando la risoluzione dei nomi di dominio di Docker Compose.

```
# If api. prefixed to ENV:APACHE_PROXY_BASE_DOMAIN equals␣
 ↪HTTP_HOST
RewriteCond "api.%{ENV:APACHE_PROXY_BASE_DOMAIN} %{HTTP_HOST}
 ↪" "^([^ ]+) \1$" [NC]
# Capture ENV:SOPHON_BACKEND_NAME for substitution in the␣
 ↪rewriterule
RewriteCond "%{ENV:SOPHON_BACKEND_NAME}" "^(.+)$" [NC]
# Forward to the backend
RewriteRule "/(.*)" "http://%1/$1" [P,L]
```

4. Carica in memoria la rubrica dei notebook generata dal *Modulo backend* e disponibile in `/run/sophon/proxy/proxy.dbm` attraverso il volume `proxy-data`, assegnandogli il nome di `sophonproxy`.

```
# Create a map between the proxy file generated by Sophon and␣
 ↪Apache
RewriteMap "sophonproxy" "dbm=gdbm:/run/sophon/proxy/proxy.dbm
 ↪"
```

5. Effettua il proxying dei websocket verso i notebook mappati dalla rubrica `sophonproxy`.

```
# If this is any other subdomain of ENV:APACHE_PROXY_BASE_
↪DOMAIN
RewriteCond ".%{ENV:APACHE_PROXY_BASE_DOMAIN} %{HTTP_HOST}" "^
↪([^ ]+) [^ ]+\1$" [NC]
# If this is a websocket connection
RewriteCond "%{HTTP:Connection}" "Upgrade" [NC]
RewriteCond "%{HTTP:Upgrade}" "websocket" [NC]
# Forward to the notebook
RewriteRule "/(.*)" "ws://${sophonproxy:%{HTTP_HOST}}/$1" [P,
↪L]
```

6. Effettua il proxying delle richieste "normali" verso i notebook mappati dalla rubrica `sophonproxy`.

```
# If this is any other subdomain of ENV:APACHE_PROXY_BASE_
↪DOMAIN
RewriteCond ".%{ENV:APACHE_PROXY_BASE_DOMAIN} %{HTTP_HOST}" "^
↪([^ ]+) [^ ]+\1$" [NC]
# Forward to the notebook
RewriteRule "/(.*)" "http://${sophonproxy:%{HTTP_HOST}}/$1"␣
↪[P,L]
```

Tutte le regole usano il flag `L` di `RewriteRule`, che porta il motore di rewriting a ignorare tutte le regole successive, come il `return` di una funzione di un linguaggio di programmazione imperativo.

5.3.1 Containerizzazione del modulo proxy

Il modulo proxy è incapsulato in un'immagine *Docker* basata sull'immagine ufficiale httpd:2.4[150], che si limita ad applicare la configurazione personalizzata.

5.4 Realizzazione del modulo Jupyter

Il *modulo Jupyter* consiste in un ambiente Jupyter[151] e JupyterLab[152] modificato per una migliore integrazione con Sophon, in particolare con il *Modulo frontend* e il *Modulo backend*.

È collocato all'interno del repository in `/jupyter`.

[150] https://hub.docker.com/_/httpd
[151] https://jupyter.org/
[152] https://jupyterlab.readthedocs.io/en/stable/

5.4.1 Sviluppo del tema per Jupyter

Per rendere l'interfaccia grafica più consistente ed user-friendly, è stato sviluppato un tema colori personalizzato per `JupyterLab`.

È stato creato partendo dal template jupyterlab/theme-cookiecutter[153], e in esso sono state modificati le variabili di stile (contenute nel file `style/variables.css`) usando i colori del tema "The Sophonity" di `Bluelib`.

È stato poi pubblicato sull'PyPI (Python Package Index) e su `npm`, permettendone l'uso a tutti gli utenti di JupyterLab.

Nota: Per facilitarne la distribuzione e il riutilizzo anche esternamente a Sophon, il tema è stato creato in un repository `Git` esterno a quello del progetto.

5.4.2 Estensione del container Docker di Jupyter

Il `Dockerfile` del modulo ne crea un `immagine Docker` in quattro fasi:

1. **Base**: Parte dall'immagine base `jupyter/scipy-notebook` e ne altera i label.

```
FROM jupyter/scipy-notebook AS base
# Set the maintainer label
LABEL maintainer="Stefano Pigozzi <me@steffo.eu>"
```

2. **Env**: Configura le variabili di ambiente dell'immagine, attivando JupyterLab, configurando il riavvio automatico di Jupyter, la collaborazione real time e permettendo all'utente non-privilegiato di acquisire i privilegi di root attraverso il comando `sudo`.

```
FROM base AS env
# Set useful envvars for Sophon notebooks
ENV JUPYTER_ENABLE_LAB=yes
ENV RESTARTABLE=yes
ENV GRANT_SUDO=yes
# Enable real time collaboration
CMD ["start-notebook.sh", "--collaborative"]
```

3. **Extensions**: Installa, abilita e configura le estensioni necessarie all'integrazione con Sophon (attualmente, soltanto il tema JupyterLab Sophon).

```
FROM env AS extensions
# As the default user...
USER ${NB_UID}
WORKDIR "${HOME}"
# Install the JupyterLab Sophon theme
RUN jupyter labextension install "jupyterlab_theme_sophon"
```

(continues on next page)

[153] https://github.com/jupyterlab/theme-cookiecutter

(continua dalla pagina precedente)

```
# Enable the JupyterLab Sophon theme
RUN jupyter labextension enable "jupyterlab_theme_sophon"
# Set the JupyterLab Sophon theme as default
RUN mkdir -p '.jupyter/lab/user-settings/@jupyterlab/apputils-
↪extension/'
RUN echo '{"theme": "JupyterLab Sophon"}' > ".jupyter/lab/
↪user-settings/@jupyterlab/apputils-extension/themes.
↪jupyterlab-settings"
```

4. **Healthcheck**: Installa curl[154], uno strumento in grado di effettuare richieste HTTP (HyperText Transfer Protocol da linea di comando, e configura la verifica dello `stato di salute` dell'immagine, al fine di comunicare al `modulo backend` il risultato di una richiesta di avvio.

```
FROM extensions AS healthcheck
# As root ...
USER root
# Install curl
RUN apt-get update
RUN apt-get install -y curl
# Use curl to check the health status
HEALTHCHECK --start-period=5s --timeout=5s --interval=10s CMD␣
↪["curl", "--output", "/dev/null", "http://localhost:8888"]

# We probably should go back to the default user
USER ${NB_UID}
```

5.5 Automazione di sviluppo

Al fine di snellire lo sviluppo del software, è stato configurato lo strumento di automazione GitHub Actions[155] per effettuare automaticamente alcuni compiti.

5.5.1 Scansione automatica delle dipendenze

È stato abilitato su *GitHub* il supporto a Dependabot[156], un software che scansiona le dipendenze dei vari moduli e notifica gli sviluppatori qualora una o più di esse siano vulnerabili ad exploit.

[154] https://curl.se/
[155] https://github.com/features/actions
[156] https://docs.github.com/en/code-security/supply-chain-security/managing-vulnerabilities-in-your-projects-dependencies/configuring-dependabot-security-updates

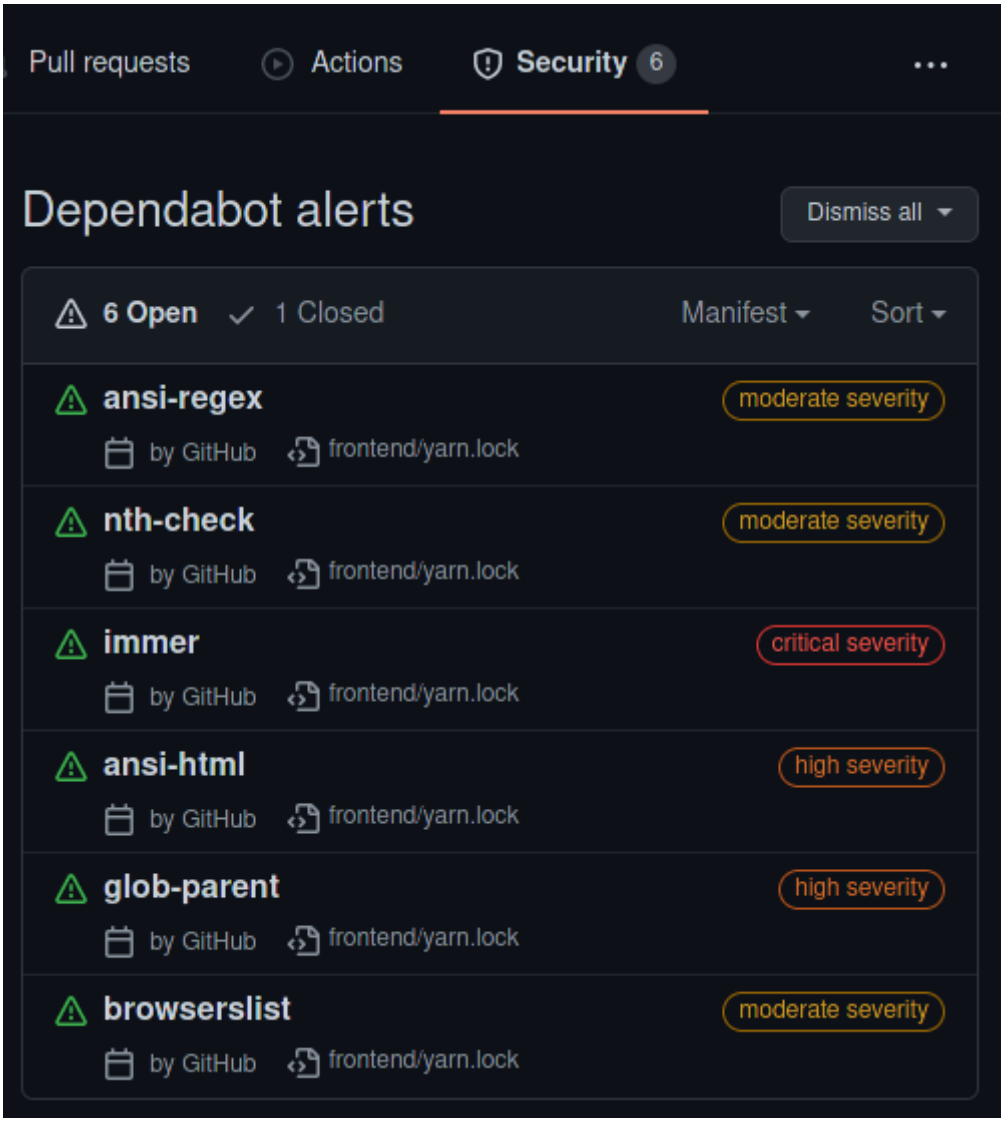


Figura 5.5.1: Alcune vulnerabilità rilevate da Dependabot all'interno delle dipendenze di Sophon.

5.5.2 Controllo automatico del codice

Sono state configurate due azioni, `analyze-codeql-backend` e `analyze-codeql-frontend`, che usano CodeQL[157] per scansionare staticamente il codice e identificare problemi o vulnerabilità.

La prima, `analyze-codeql-backend`, viene eseguita solo quando viene inviato a GitHub nuovo codice relativo al `modulo backend`, ed effettua analisi specifiche a `Python`, mentre la seconda, `analyze-codeql-frontend`, viene eseguita solo quando viene inviato nuovo codice del `modulo frontend`, ed effettua analisi specifiche a JavaScript.

Si riportano due estratti relativi all'azione `analyze-codeql-backend`.

```yaml
on:
   push:
      branches: [ main ]
      paths:
         - "backend/**"
```

```yaml
steps:
   - name: Checkout repository
     uses: actions/checkout@v2
   - name: Initialize CodeQL
     uses: github/codeql-action/init@v1
     with:
       languages: "python"
   - name: Perform CodeQL Analysis
     uses: github/codeql-action/analyze@v1
```

5.5.3 Costruzione automatica delle immagini Docker

Sono state configurate quattro azioni, `build-docker-frontend`, `build-docker-backend`, `build-docker-jupyter` e `build-docker-proxy`, che costruiscono automaticamente l'immagine *Docker* di ciascun modulo qualora il relativo codice venga modificato.

L'immagine creata viene poi caricata sul GitHub Container Registry[158], da cui può poi essere scaricata attraverso *Docker*.

Si riporta un estratto relativo all'azione `build-docker-proxy`.

```yaml
steps:
   - name: "Checkout repository"
     uses: actions/checkout@v2
   - name: "Login to GitHub Containers"
     run: echo ${{ secrets.GITHUB_TOKEN }} | docker login ghcr.io -
↪u Steffo99 --password-stdin
```

(continues on next page)

[157] https://codeql.github.com/

[158] https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry

(continua dalla pagina precedente)

```
   - name: "Build the docker container `ghcr.io/steffo99/sophon-
↪proxy:latest`"
     run: docker build ./proxy --tag ghcr.io/steffo99/sophon-
↪proxy:latest
   - name: "Upload the container to GitHub Containers"
     run: docker push ghcr.io/steffo99/sophon-proxy:latest
```

5.5.4 Costruzione automatica della documentazione

Sono state configurate due azioni, `build-sphinx-report` e `build-sphinx-thesis`, che compilano rispettivamente la documentazione richiesta per l'esame di Tecnologie Web e questa stessa tesi usando lo strumento Sphinx[159].

La documentazione per l'esame viene compilata solo da reStructuredText[160] ad HTML; la tesi, invece, viene compilata sia in HTML sia in PDF.

Si riporta un estratto relativo all'azione `build-sphinx-thesis`.

```
latexpdf:
   name: "Build PDF document"
   runs-on: ubuntu-latest
   steps:
      - name: "Update apt repositories"
        run: sudo apt-get update -y
      - name: "Checkout repository"
        uses: actions/checkout@v2
        with:
           lfs: true
      - name: "Checkout LFS objects"
        run: git lfs checkout
      - name: "Setup Python"
        uses: actions/setup-python@v2
        with:
           python-version: 3.9
      - name: "Setup Poetry"
        uses: abatilo/actions-poetry@v2.0.0
        with:
           poetry-version: 1.1.11
      - name: "Install LaTeX packages"
        run: sudo apt-get install -y latexmk texlive-latex-
↪recommended texlive-latex-extra texlive-fonts-recommended␣
↪texlive-luatex fonts-ebgaramond fonts-ebgaramond-extra fonts-
↪firacode xindy
      - name: "Install backend dependencies"
        working-directory: backend/
```

(continues on next page)

[159] https://www.sphinx-doc.org/en/master/
[160] https://docutils.sourceforge.io/rst.html

(continua dalla pagina precedente)

```
        run: poetry install --no-interaction
      - name: "Find Poetry Python environment"
        working-directory: backend/
        run: echo "pythonLocation=$(poetry env list --full-path |␣
↪cut -f1 -d' ')/bin" >> $GITHUB_ENV
      - name: "Build LaTeX document with Sphinx"
        working-directory: thesis/
        run: |
           source $pythonLocation/activate
           make latexpdf
      - name: "Upload build artifact"
        uses: actions/upload-artifact@v2
        with:
           name: "thesis.pdf"
           path: "thesis/build/latex/
↪progettazioneesviluppodisophonapplicativocloudasupportodellaricerca.
↪pdf"
```

CAPITOLO 6

Risultati ottenuti

Al termine del periodo di sviluppo, il software ha soddisfatto *tutti i requisiti prefissati*.

In particolare:

- tutte le funzionalità desiderate sono state sviluppate, raggiungendo la feature parity con JupyterHub;
  - il software ha ampio spazio per eventuali *estensioni* grazie alle numerosi classi astratte sviluppate;
  - l'isolamento tra notebook e l'autenticazione all'accesso è stata realizzata, garantendo la *sicurezza* dei dati;
  - l'interfaccia web è sufficientemente *intuitiva* per permetterne un utilizzo e apprendimento autonomo;
  - è possibile *personalizzare* i dettagli del software con il brand della propria istituzione;
  - più utenti possono *collaborare* simultaneamente all'interno dello stesso notebook;
  - il software è stato pubblicato su *GitHub* come progetto *open source*;
  - da dispositivi mobili, l'interfaccia grafica di Sophon risulta interamente utilizzabile, raggiungendo il requisito di *responsività*, anche se la modifica di notebook computazionali con JupyterLab potrebbe risultare difficile su schermi con risoluzione molto ridotta;
  - l'interfaccia web soddisfa i requisiti di *accessibilità* fissati;
- il software è sufficientemente stabile per l'utilizzo in produzione, permettendone il suo utilizzo all'interno dell'Università;
- le *istruzioni per l'installazione* sono state scritte, permettendo ad altri interessati di installare Sophon.

6.1 Stato finale del modulo backend

Il modulo backend terminato espone una web API all'indirizzo `api.BASE_DOMAIN` con i seguenti endpoints.

`POST /api/auth/token/`

Effettua l'accesso in cambio di un token di autenticazione.

JSON Parameters

- **`username`** (`string`) -- Username dell'utente.
- **`password`** (`string`) -- Password dell'utente.

Status Codes

- 200 OK[161] -- Login riuscito.

`POST /api/auth/session/`

Effettua l'accesso, salvando i dati di autenticazione nei cookie.

JSON Parameters

- **`username`** (`string`) -- Username dell'utente.
- **`password`** (`string`) -- Password dell'utente.

Status Codes

- 200 OK[162] -- Login riuscito.

`ANY /api/core/groups/`

Accede ai *gruppi di ricerca*, permettendone la visualizzazione (`GET`), la creazione (`POST`), la modifica (`PUT`) e l'eliminazione (`DELETE`), a condizione che si sia autorizzati ad effettuare l'operazione.

JSON Parameters

- **`slug`** (`string`) -- Slug del gruppo di ricerca.
- **`name`** (`string`) -- Nome del gruppo di ricerca.
- **`description`** (`string`) -- Descrizione del gruppo di ricerca.
- **`access`** (`string`) -- *Modalità di accesso* al gruppo.
- **`owner`** (`integer`) -- ID del creatore del gruppo.
- **`members`** (`integer[]`) -- Elenco dei membri degli ID dei membri del gruppo.

Status Codes

- 200 OK[163] -- Operazione effettuata.
- 201 Created[164] -- Risorsa creata.

[161] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[162] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1

- 204 No Content[165] -- Risorsa eliminata.
- 401 Unauthorized[166] -- Accesso non effettuato.
- 403 Forbidden[167] -- Operazione non permessa.
- 404 Not Found[168] -- Risorsa non esistente.

`GET /api/core/users/by-id/`

Accede agli *utenti* dell'istanza Sophon usando il loro ID come chiave, permettendone la visualizzazione.

JSON Parameters

- **`id`** (`integer`) -- ID dell'utente.
- **`username`** (`string`) -- Username dell'utente.
- **`first_name`** (`string`) -- Nome dell'utente (non utilizzato se non specificato manualmente nell'interfaccia di amministrazione).
- **`last_name`** (`string`) -- Cognome dell'utente (non utilizzato se non specificato manualmente nell'interfaccia di amministrazione).
- **`email`** (`string`) -- Email dell'utente (non utilizzata se non specificata manualmente nell'interfaccia di amministrazione).

Status Codes

- 200 OK[169] -- Operazione effettuata.

`GET /api/core/users/by-username/`

Accede agli *utenti* dell'istanza Sophon usando il loro username come chiave, permettendone la visualizzazione.

JSON Parameters

- **`id`** (`string`) -- ID dell'utente.
- **`username`** (`string`) -- Username dell'utente.
- **`first_name`** (`string`) -- Nome dell'utente (non utilizzato se non specificato manualmente nell'interfaccia di amministrazione).
- **`last_name`** (`string`) -- Cognome dell'utente (non utilizzato se non specificato manualmente nell'interfaccia di amministrazione).
- **`email`** (`string`) -- Email dell'utente (non utilizzata se non specificata manualmente nell'interfaccia di amministrazione).

Status Codes

[163] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[164] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[165] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[166] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[167] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.4
[168] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[169] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1

- 200 OK[170] -- Operazione effettuata.

ANY /api/projects/by-slug/

Accede a tutti i *progetti di ricerca* dell'istanza Sophon, permettendone la visualizzazione (GET), la creazione (POST), la modifica (PUT) e l'eliminazione (DELETE), a condizione che si sia autorizzati ad effettuare l'operazione.

JSON Parameters

- **slug** (string) -- Slug del progetto.
- **name** (string) -- Nome del progetto.
- **description** (string) -- Descrizione del progetto.
- **visibility** (string) -- *Visibilità* del progetto.
- **group** (string) -- Slug del gruppo a cui appartiene il progetto.

Status Codes

- 200 OK[171] -- Operazione effettuata.
- 201 Created[172] -- Risorsa creata.
- 204 No Content[173] -- Risorsa eliminata.
- 401 Unauthorized[174] -- Accesso non effettuato.
- 403 Forbidden[175] -- Operazione non permessa.
- 404 Not Found[176] -- Risorsa non esistente.

ANY /api/projects/by-group/(str: *group_slug*)/

Accede ai *progetti di ricerca* appartenenti a un certo gruppo, permettendone la visualizzazione (GET), la creazione (POST), la modifica (PUT) e l'eliminazione (DELETE), a condizione che si sia autorizzati ad effettuare l'operazione.

Parameters

- **group_slug** -- Slug del gruppo di cui si vogliono ottenere i progetti.

JSON Parameters

- **slug** (string) -- Slug del progetto.
- **name** (string) -- Nome del progetto.
- **description** (string) -- Descrizione del progetto.
- **visibility** (string) -- *Visibilità* del progetto.
- **group** (string) -- Slug del gruppo a cui appartiene il progetto.

[170] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[171] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[172] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[173] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[174] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[175] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.4
[176] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5

Status Codes

- 200 OK[177] -- Operazione effettuata.
- 201 Created[178] -- Risorsa creata.
- 204 No Content[179] -- Risorsa eliminata.
- 401 Unauthorized[180] -- Accesso non effettuato.
- 403 Forbidden[181] -- Operazione non permessa.
- 404 Not Found[182] -- Risorsa non esistente.

`ANY /api/notebooks/by-slug/`

Accede a tutti i *notebook* dell'istanza Sophon, permettendone la visualizzazione (`GET`), la creazione (`POST`), la modifica (`PUT`) e l'eliminazione (`DELETE`), a condizione che si sia autorizzati ad effettuare l'operazione.

Nota: Questo endpoint non restituisce i dettagli di connessione al notebook; a tale scopo, è necessario utilizzare `ANY /api/notebooks/by-project/(str:project_slug)/`.

JSON Parameters

- **`slug`** (`string`) -- Slug del notebook.
- **`name`** (`string`) -- Nome del notebook.
- **`is_running`** (`boolean`) -- Se il notebook è *avviato* oppure no.
- **`locked_by`** (`integer`) -- ID dell'utente che ha *bloccato* il notebook.
- **`container_image`** (`string`) -- Il nome dell'*immagine* del notebook.
- **`project`** (`string`) -- Slug del progetto a cui appartiene il notebook.

Status Codes

- 200 OK[183] -- Operazione effettuata.
- 201 Created[184] -- Risorsa creata.
- 204 No Content[185] -- Risorsa eliminata.
- 401 Unauthorized[186] -- Accesso non effettuato.
- 403 Forbidden[187] -- Operazione non permessa.
- 404 Not Found[188] -- Risorsa non esistente.

[177] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[178] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[179] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[180] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[181] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.4
[182] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5

ANY /api/notebooks/by-project/(str: *project_slug*)/

Accede ai *notebook* appartenenti a un certo progetto, permettendone la visualizzazione (GET), la creazione (POST), la modifica (PUT) e l'eliminazione (DELETE), a condizione che si sia autorizzati ad effettuare l'operazione.

JSON Parameters

- **slug** (string) -- Slug del notebook.
- **name** (string) -- Nome del notebook.
- **is_running** (boolean) -- Se il notebook è *avviato* oppure no.
- **locked_by** (integer) -- ID dell'utente che ha *bloccato* il notebook.
- **container_image** (string) -- Il nome dell'*immagine* del notebook.
- **project** (string) -- Slug del progetto a cui appartiene il notebook.
- **jupyter_token** (string) -- Token per l'autenticazione sul *modulo Jupyter*.
- **legacy_notebook_url** (string) -- URL per la connessione all'interfaccia legacy "*Jupyter Notebook*" del notebook.
- **lab_url** (string) -- URL per la connessione all'interfaccia *JupyterLab* del notebook.

Status Codes

- 200 OK[189] -- Operazione effettuata.
- 201 Created[190] -- Risorsa creata.
- 204 No Content[191] -- Risorsa eliminata.
- 401 Unauthorized[192] -- Accesso non effettuato.
- 403 Forbidden[193] -- Operazione non permessa.
- 404 Not Found[194] -- Risorsa non esistente.

[183] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[184] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[185] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[186] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[187] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.4
[188] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5
[189] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[190] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2
[191] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.5
[192] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.2
[193] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.4
[194] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5

6.1.1 Pagina di amministrazione esposta

In aggiunta alla web API, Sophon espone la *pagina di amministrazione* Django al seguente URL.

`GET /admin/`
La pagina di amministrazione Django, personalizzata per Sophon.

Status Codes

- 200 OK[195] -- Accesso riuscito.

La prima pagina richiede l'accesso con credenziali di un *superutente*.

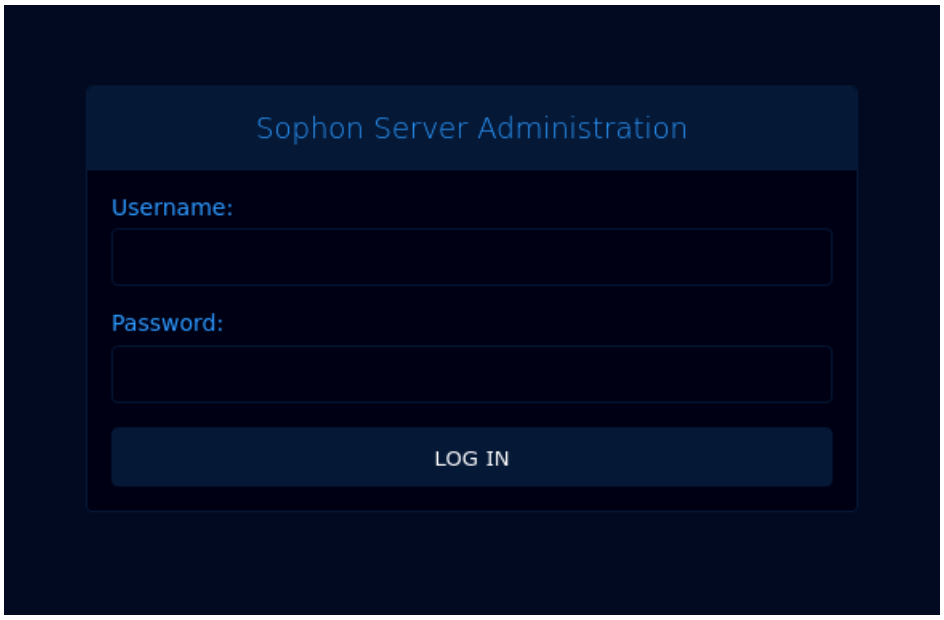


Figura 6.1.1: Schermata di login della pagina di amministrazione.

Una volta effettuato l'accesso, all'interno della pagina è possibile modificare ogni genere di *entità* presente nell'istanza.

6.2 Stato finale del modulo frontend

Il modulo frontend terminato espone una SPA (single page app) all'indirizzo `BASE_DOMAIN`.

[195] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1

Figura 6.1.2: Elenco delle entità presenti all'interno dell'istanza.

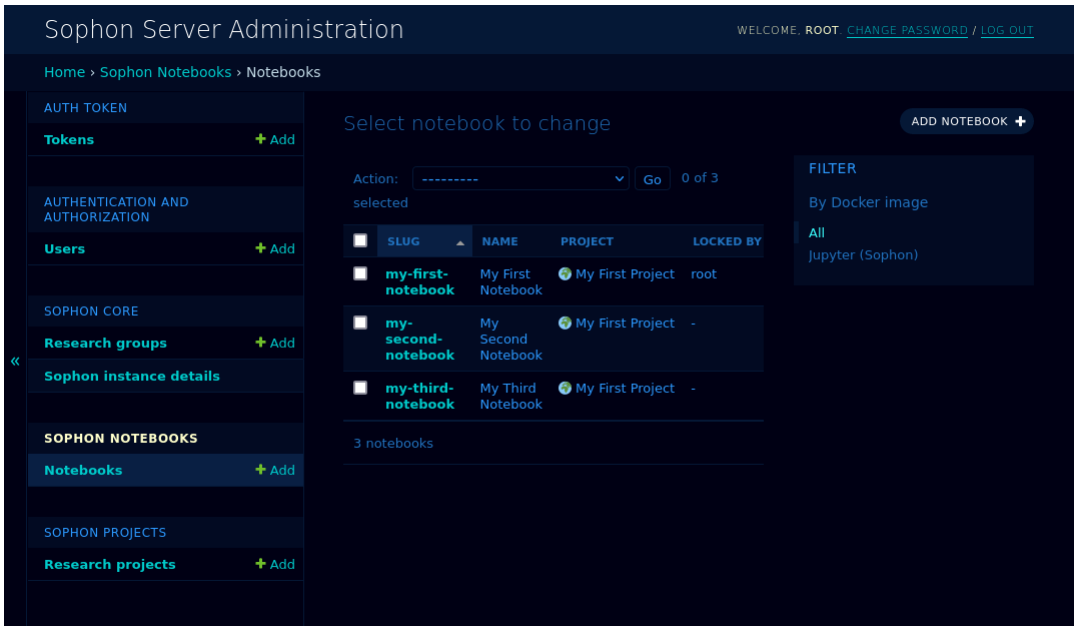


Figura 6.1.3: Elenco dei notebook presenti all'interno dell'istanza di dimostrazione.

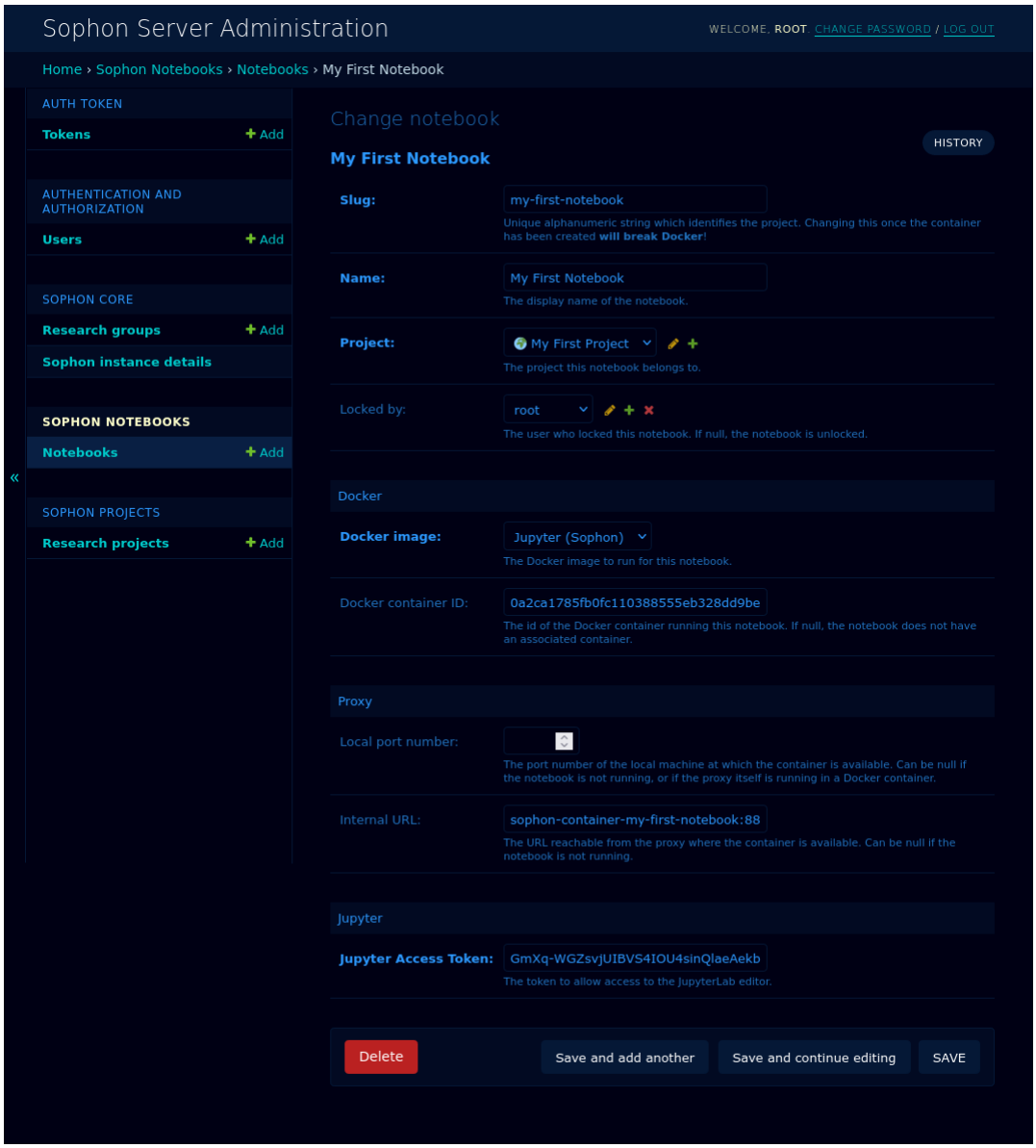


Figura 6.1.4: Pagina di modifica di uno dei notebook dell'istanza di dimostrazione.

Figura 6.1.5: Pagina di modifica dei dettagli dell'istanza Sophon.

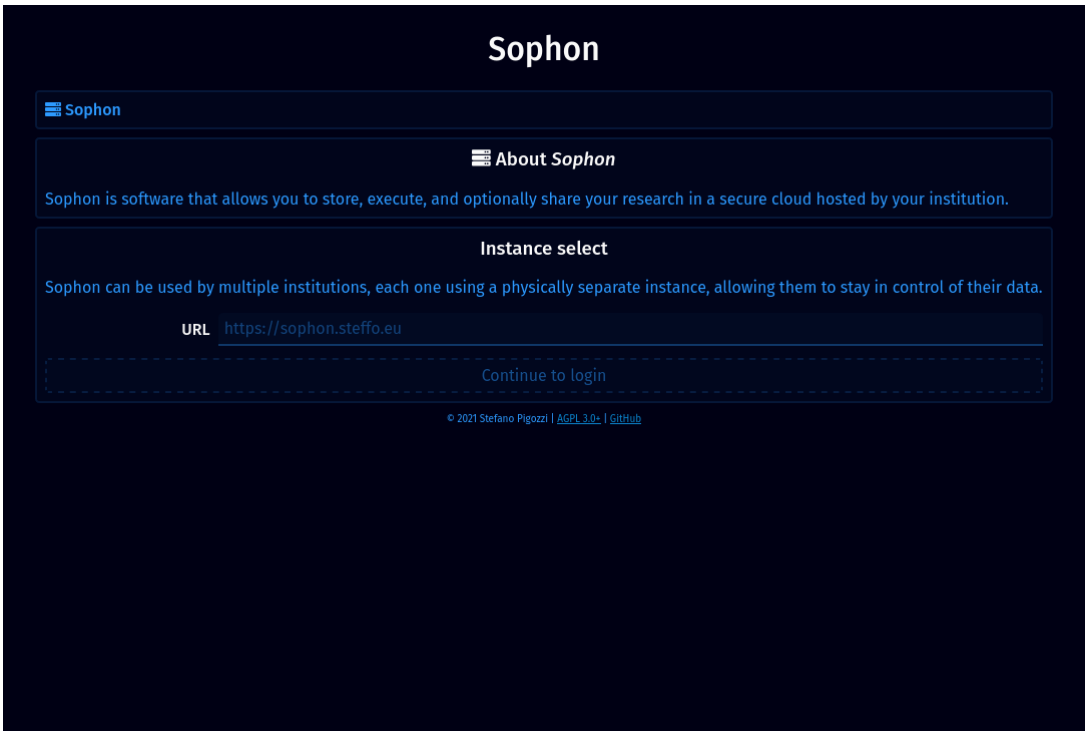


Figura 6.2.1: Pagina di selezione istanza.

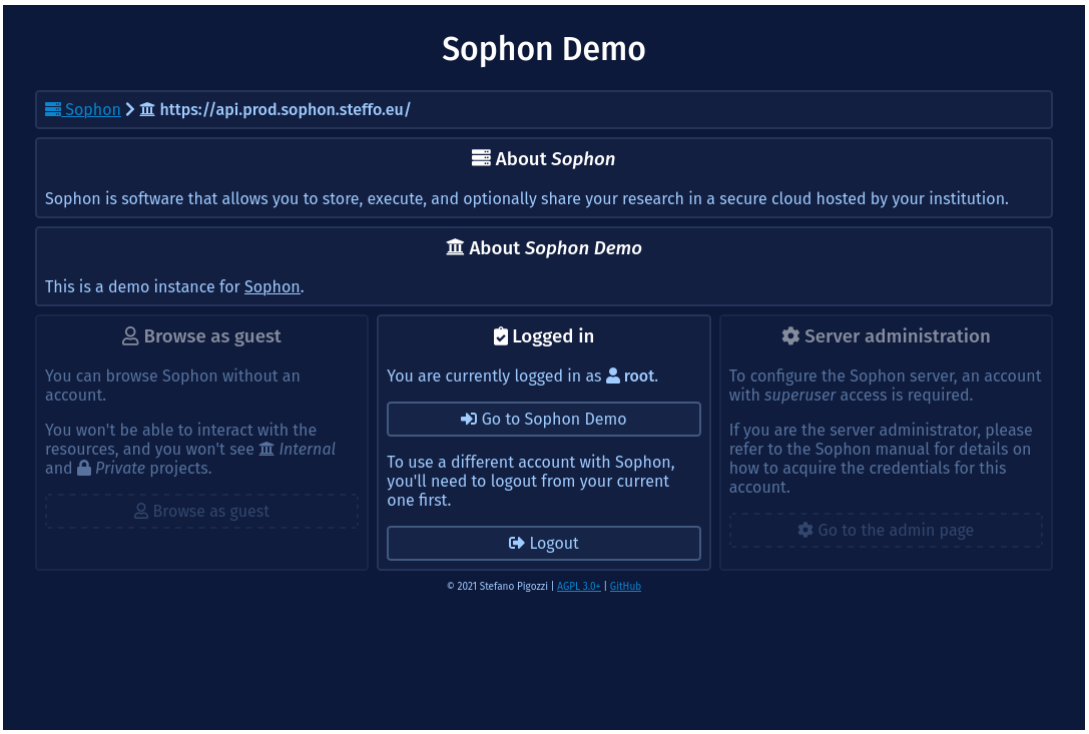


Figura 6.2.2: Pagina di login all'istanza Sophon di dimostrazione, che utilizza il tema "Royal Blue".

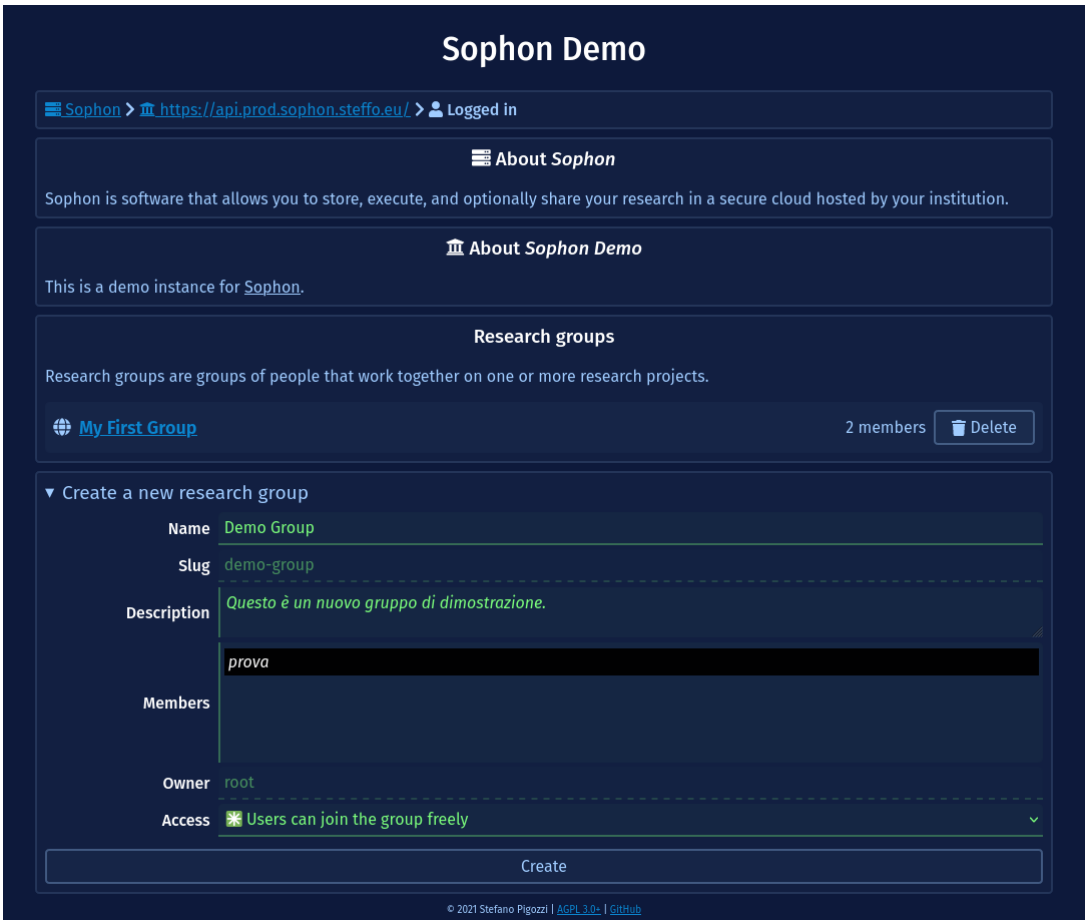


Figura 6.2.3: Pagina di selezione e creazione gruppi di ricerca.

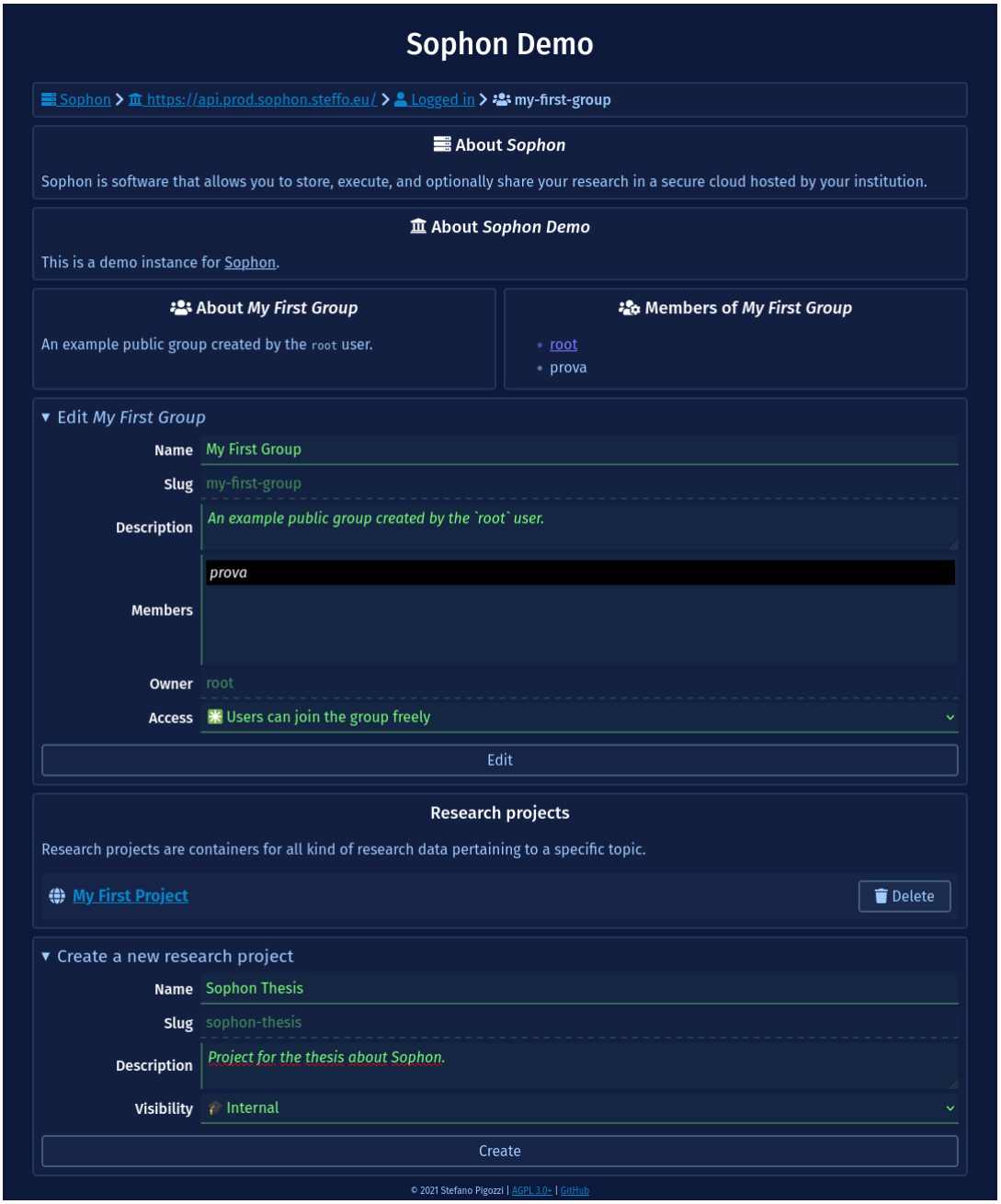


Figura 6.2.4: Pagina di selezione e creazione progetti di ricerca.

Figura 6.2.5: Pagina di selezione, creazione e avvio notebook.

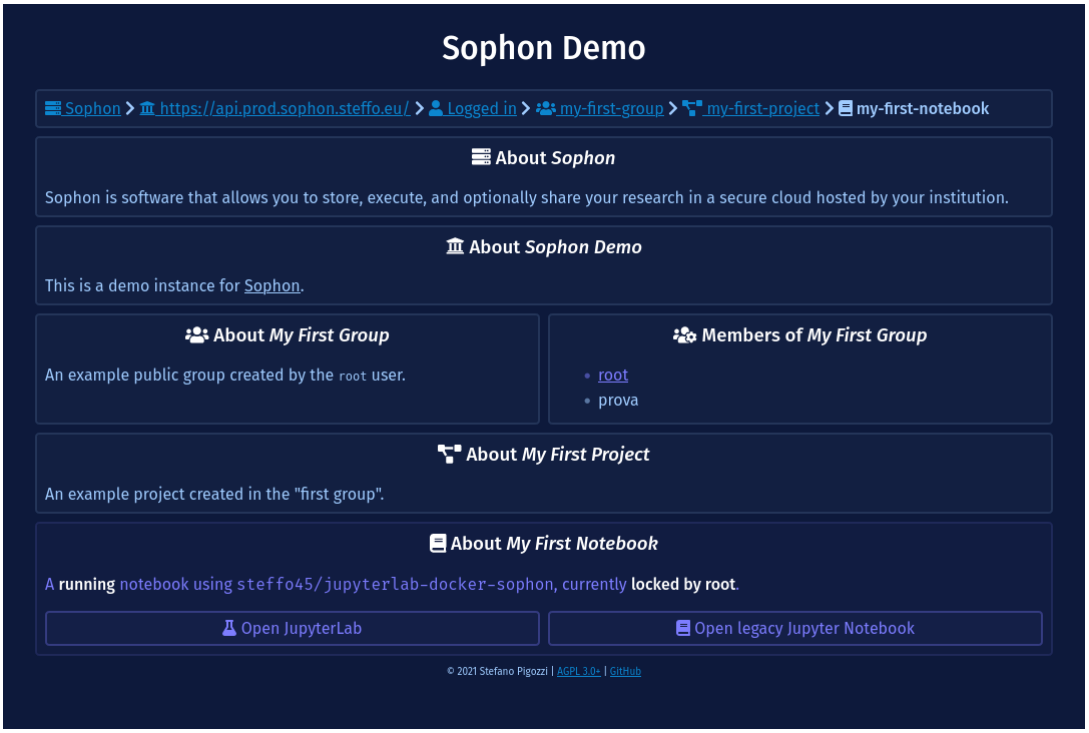
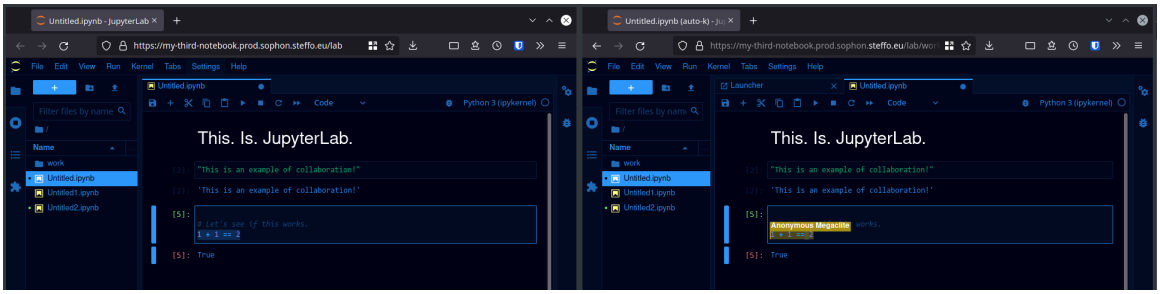


Figura 6.2.6: Pagina di dettagli di un notebook, che permette l'accesso al modulo Jupyter.

6.3 Stato finale del modulo Jupyter

Il modulo Jupyter terminato espone un'istanza collaborativa di *JupyterLab* all'indirizzo `NOTEBOOK_SLUG.BASE_DOMAIN`.



6.4 Stato finale del modulo proxy

Il modulo proxy terminato effettua correttamente proxying tra gli altri moduli.

CAPITOLO 7

Il futuro di Sophon

Lo sviluppo di Sophon lascia aperte innumerevoli strade per la sua espansione con funzionalità aggiuntive.

Si conclude la tesi analizzandone alcune.

7.1 Repository GitHub di Sophon

È stato creato un repository per il progetto su GitHub[196].

Oltre al codice sorgente, esso include un issue tracker[197], all'interno del quale viene tenuto traccia di tutte le proposte di funzionalità aggiuntive.

Si elencano alcune delle funzionalità proposte.

7.1.1 Nuova entità: il documento

Si propone di sviluppare una nuova entità, il *documento*, che permetterebbe agli utenti di Sophon di creare testi in Markdown senza uscire dall'interfaccia web, e di renderli disponibili basandosi sul sistema di permessi di Sophon.

[196] https://github.com/Steffo99/sophon
[197] https://github.com/Steffo99/sophon/issues

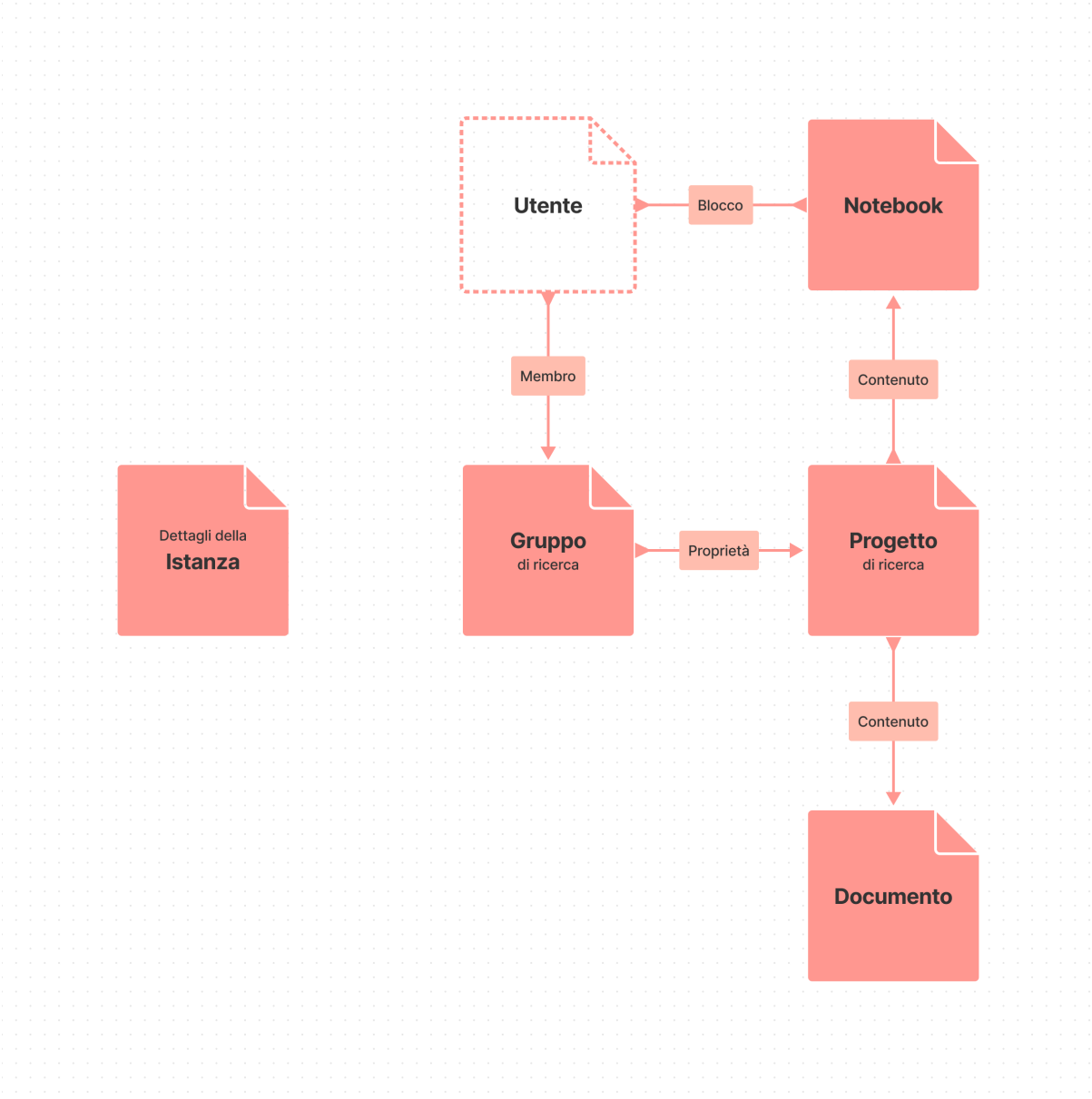


Figura 7.1.1: Schema del database se venisse aggiunta l'entità "Documento".

7.1.2 Sistema per organizzazione delle entità

Si propone di realizzare dei sistemi che permettano di catalogare e raggruppare le entità di ogni tipo attraverso parole chiave (*tag*) selezionabili dal creatore della relativa entità.

Un esempio di tag potrebbe essere `[Tesi]`, utilizzabile per i progetti relativi alle tesi degli studenti di un corso.

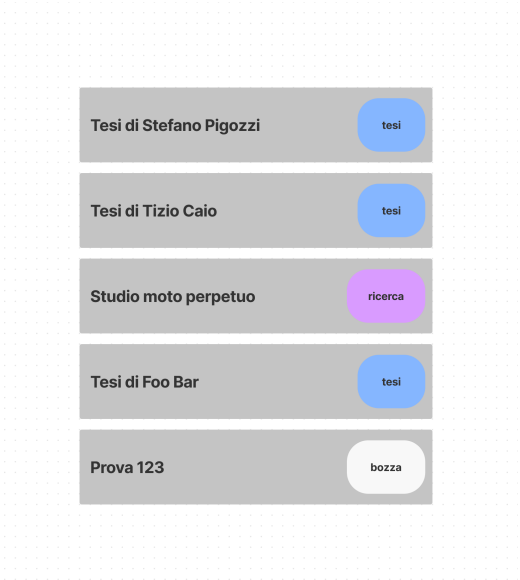


Figura 7.1.2: Un esempio di come potrebbero funzionare i tag applicati ai progetti.

7.1.3 Registro delle attività

Si propone di creare un registro, detto *delle attività* o in inglese *activity log*, all'interno del quale siano registrate tutte le azioni effettuate sulle entità del progetto.

Ciò favorirebbe la accountability tra gli utenti di Sophon, in quanto diverrebbe possibile identificare il responsabile di certe azioni distruttive, come l'eliminazione di un intero gruppo.

7.1.4 Federazione tra istanze

L'ultima proposta, molto ambiziosa, sarebbe quella di permettere la *federazione* tra le varie istanze Sophon, consentendo la condivisione di risorse attraverso più istituzioni senza dover creare utenti "locali" per ciascun collaboratore.

Figura 7.1.3: Un esempio di come potrebbe funzionare il registro delle attività.

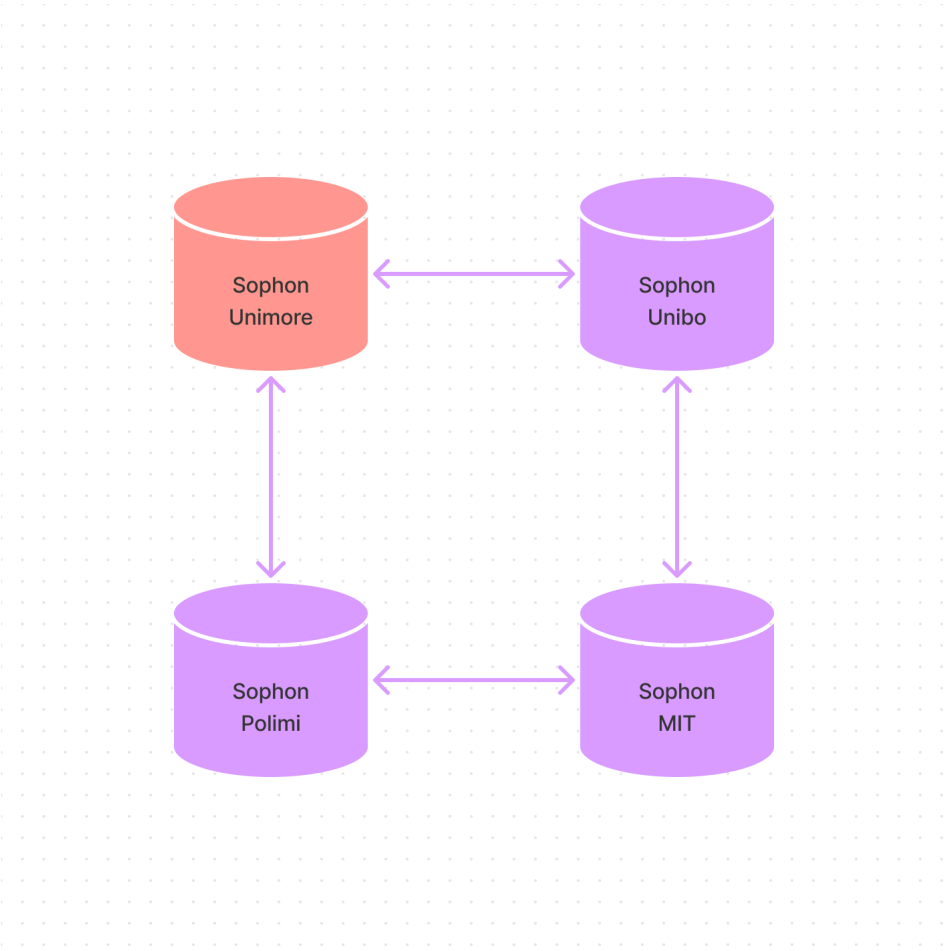


Figura 7.1.4: Un diagramma di esempio di possibile federazione di Sophon.

CAPITOLO 8

Installazione di Sophon

Questo capitolo descrive la procedura da seguire per installare Sophon.

8.1 Requisiti dell'host

- Una connessione ad Internet (solo in fase di installazione)
- GNU Wget[198] (solo in fase di installazione)
- Un nome di dominio
- Un webserver (ad esempio, Apache HTTPd[199])
- Un certificato SSL valido *(opzionale, ma raccomandato)*
- Docker Engine[200]
- Docker Compose[201]

Suggerimento: È possibile ottenere gratuitamente un certificato SSL utilizzando Letsencrypt[202]!

[198] https://www.gnu.org/software/wget/
[199] https://httpd.apache.org/
[200] https://docs.docker.com/engine/
[201] https://docs.docker.com/compose/
[202] https://letsencrypt.org/

8.2 Preparazione di Docker Compose

Come `root`, si crei una nuova cartella sul proprio sistema operativo in cui archiviare le risorse relative a Sophon:

```
root:~# mkdir -p /dock/sophon
```

Successivamente, si scarichi il file `docker-compose.yml` all'interno della cartella dal repository di Sophon:

```
root:~# cd /dock/sophon

root:/dock/sophon# wget "https://raw.githubusercontent.com/
↪Steffo99/sophon/main/docker-compose.yml"
--2021-11-02 18:03:05--  https://raw.githubusercontent.com/
↪Steffo99/sophon/main/docker-compose.yml
SSL_INIT
Loaded CA certificate '/etc/ssl/certs/ca-certificates.crt'
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... ␣
↪185.199.108.133,
185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.
↪com) 185.199.108.133:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 2957 (2.9K) [text/plain]
Saving to: ‘docker-compose.yml’

docker-compose.yml  100%[═══════════════════⇒]   2.89K  --.-KB/s  ␣
↪ in 0s

2021-11-02 18:03:05 (48.0 MB/s) - ‘docker-compose.yml’ saved [2957/
↪2957]
```

8.3 Configurazione DNS

Si scelga il dominio (o sottodominio) sul quale si vuole che Sophon sia accessibile e si aggiungano i seguenti record DNS, sostituendo il dominio `ilmiosophon.it` con il proprio nome di dominio, e gli indirizzi IPv4 e IPv6 del server al posto di `0.0.0.0` e `1234::1234`:

```
*.ilmiosophon.it 1800 IN A    0.0.0.0
*.ilmiosophon.it 1800 IN AAAA 1234::1234
ilmiosophon.it   1800 IN A    0.0.0.0
ilmiosophon.it   1800 IN AAAA 1234::1234
```

Sophon sarà quindi accessibile ai seguenti indirizzi:

- l'interfaccia web al dominio base (`https://ilmiosophon.it/`);

- l'API al dominio base prefisso con `api.` (`https://api.ilmiosophon.it/`);
- i notebook al dominio base prefissi con lo slug del notebook (`https://ilmionotebook.ilmiosophon.it/`).

8.4 Configurazione `docker-compose.yml`

Si configuri con l'editor di testo preferito il file `docker-compose.yml` con le impostazioni desiderate.

```
root:/dock/sophon# open docker-compose.yml
```

In particolare, tutte le impostazioni precedute da `# INSTALL` vanno obbligatoriamente modificate.

`DJANGO_SECRET_KEY`

Specifica la chiave segreta da usare per i cookie di sessione.

```
- DJANGO_SECRET_KEY=do-not-use-this-key-in-production-or-you-
↪will-get-hacked
```

> **Avvertimento:** Cambiare la chiave segreta una volta installato Sophon invaliderà tutti gli accessi effettuati dagli utenti.

> **Pericolo:** La chiave segreta è un dato estremamente riservato: chiunque sia a conoscenza della chiave segreta potrà effettuare l'accesso come qualsiasi utente!

Vedi anche:

SECRET_KEY[203] nella documentazione di Django.

`DJANGO_PROXY_BASE_DOMAIN`

Specifica il dominio che dovrà essere usato come radice per il proxy, ovvero il dominio per il quale si è configurato il DNS in precedenza.

Se non è specificato, Sophon verrà eseguito in *modalità sviluppo*.

```
- DJANGO_PROXY_BASE_DOMAIN=ilmiosophon.it
```

`DJANGO_PROXY_PROTOCOL`

Specifica il protocollo che dovrà essere usato nei mapping del proxy.

Si consiglia di utilizzare `https`, ma è un valore valido anche `http`.

[203] https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-SECRET_KEY

```
- DJANGO_PROXY_PROTOCOL=https
```

DJANGO_ALLOWED_HOSTS

Specifica i domini da cui possono provenire le richieste alla pagina di amministrazione.

Per specificare più domini, è necessario separarli con dei pipe `|`.

Eccetto in configurazioni speciali, deve essere uguale al dominio prefisso da `api.`.

```
- DJANGO_ALLOWED_HOSTS=api.ilmiosophon.it
```

Vedi anche:

ALLOWED_HOSTS[204] nella documentazione di Django.

DJANGO_ALLOWED_ORIGINS

Specifica i domini da cui possono provenire le richieste all'API.

Per specificare più domini, è necessario separarli con dei pipe `|`.

Eccetto in configurazioni speciali, deve contenere il proprio dominio prefisso dal protocollo, e in aggiunta il dominio speciale `https://sophon.steffo.eu`, necessario per permettere l'accesso dall'interfaccia web "universale" di Sophon.

```
- DJANGO_ALLOWED_ORIGINS=https://ilmiosophon.it|https:/
↪/sophon.steffo.eu
```

Vedi anche:

L'header Access-Control-Allow-Origin[205] su MDN.

DJANGO_STATIC_URL

Specifica l'URL a cui saranno accessibili i file statici di Sophon.

Eccetto in configurazioni speciali, deve essere uguale alla seguente stringa, con le parole in maiuscolo sostituite rispettivamente dal protocollo e dal dominio selezionato: `PROTOCOLLO://static.DOMINIO/django-static/`.

```
- DJANGO_ALLOWED_ORIGINS=http://static.ilmiosophon.it/django-
↪static/
```

> **Avvertimento:** Ci si assicuri che sia presente uno slash al termine della stringa, oppure il pannello di amministrazione non sarà visualizzato correttamente!

Vedi anche:

STATIC_URL[206] nella documentazione di Django

[204] https://docs.djangoproject.com/en/3.2/ref/settings/#allowed-hosts
[205] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin
[206] https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-STATIC_URL

DJANGO_LANGUAGE_CODE

Specifica la lingua che deve usare Sophon nei messaggi di errore.

Usa il formato language code[207] di Django.

```
- DJANGO_LANGUAGE_CODE=en-us
```

Vedi anche:

LANGUAGE_CODE[208] nella documentazione di Django

DJANGO_TIME_ZONE

Specifica il fuso orario che deve usare Sophon nell'interfaccia di amministrazione.

Usa il formato tzdata[209].

```
- DJANGO_TIME_ZONE=Europe/Paris
```

Suggerimento: Il fuso orario italiano è `Europe/Rome`.

DJANGO_SU_USERNAME

Specifica il nome del *superutente* che verrà automaticamente creato qualora il database non contenga altri utenti.

```
- DJANGO_SU_USERNAME=root
```

DJANGO_SU_EMAIL

Specifica l'email del *superutente* che verrà automaticamente creato qualora il database non contenga altri utenti.

```
- DJANGO_SU_USERNAME=django@example.org
```

Nota: Attualmente, l'email non è utilizzata, ma è richiesta da Django per la creazione di un nuovo utente.

DJANGO_SU_PASSWORD

Specifica la password del *superutente* che verrà automaticamente creato qualora il database non contenga altri utenti.

```
- DJANGO_SU_PASSWORD=square
```

Avvertimento: La password è un dato estremamente riservato, in quanto chiunque ne venga a conoscenza potrà accedere a Sophon con pieni privilegi!

[207] https://docs.djangoproject.com/en/3.2/topics/i18n/#term-language-code
[208] https://docs.djangoproject.com/en/3.2/ref/settings/#language-code
[209] https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

`REACT_APP_DEFAULT_INSTANCE`
: Specifica il valore con cui precompilare il campo "selezione istanza" dell'interfaccia web di Sophon.

  Eccetto in configurazioni speciali, deve essere uguale al dominio prefisso dal protocollo e da `api.`.

```
- REACT_APP_DEFAULT_INSTANCE=https://api.ilmiosophon.it
```

`APACHE_PROXY_BASE_DOMAIN`
: Specifica il dominio che dovrà essere usato come radice per il proxy, ovvero il `DOMINIO` per il quale si è configurato il DNS in precedenza.

  Deve essere uguale a `DJANGO_PROXY_BASE_DOMAIN`.

```
- APACHE_PROXY_BASE_DOMAIN=ilmiosophon.it
```

8.5 Download delle immagini Docker

Si utilizzi Docker Compose[210] per scaricare le immagini[211] Docker necessarie all'avvio di Sophon:

```
root:/dock/sophon# docker compose pull
[+] Running 4/4
 ⠿ proxy Pulled                                                    ⌐
↪                                  1.5s
 ⠿ frontend Pulled                                                 ⌐
↪                                  1.4s
 ⠿ db Pulled                                                       ⌐
↪                                  1.9s
 ⠿ backend Pulled                                                  ⌐
↪                                  1.6s
```

Inoltre, si scarichi manualmente l'*Immagine del notebook* che può essere avviata da Sophon:

```
root:/dock/sophon# docker image pull "ghcr.io/steffo99/sophon-
↪jupyter:latest"
latest: Pulling from steffo99/sophon-jupyter
7b1a6ab2e44d: Already exists
578d7ac380c6: Pull complete
37f1e0b584f6: Pull complete
3c7282703390: Pull complete
b38aa558f711: Pull complete
1412103d568f: Pull complete
67419a9a821e: Pull complete
37e6cc015184: Pull complete
7d9316e2b57c: Pull complete
```

(continues on next page)

[210] https://docs.docker.com/compose/
[211] https://docs.docker.com/engine/reference/commandline/images/

(continua dalla pagina precedente)

```
a7f024508c72: Pull complete
f3eae3c301a1: Pull complete
d3e2107efade: Pull complete
d94bc6f8f069: Pull complete
1e1dc3e818ad: Pull complete
c975ee664182: Pull complete
101cfcc0e15b: Pull complete
bf991a0d7538: Pull complete
4c044af18c7e: Pull complete
605d8c6e8eba: Pull complete
ed06f2ae4a88: Pull complete
ed8b1c841d10: Pull complete
468fe9a390ae: Pull complete
Digest: sha256:5d42e5e40e406130c688914d6a58aa94769eab03620b53e0fd40
 ↪9a7fb2682a01
Status: Downloaded newer image for ghcr.io/steffo99/sophon-
 ↪jupyter:latest
ghcr.io/steffo99/sophon-jupyter:latest
```

8.6 Avvio di Sophon

Si utilizzi Docker Compose[212] per eseguire le immagini[213] di Sophon precedentemente scaricate:

```
root:/dock/sophon# docker compose up -d
[+] Running 4/4
 ⠿ Container sophon-db-1        Started                                 ⌐
 ↪                             11.3s
 ⠿ Container sophon-frontend-1  Started                                 ⌐
 ↪                             11.7s
 ⠿ Container sophon-backend-1   Started                                 ⌐
 ↪                             10.1s
 ⠿ Container sophon-proxy-1     Started                                 ⌐
 ↪                             11.5s
```

Si verifichi che i container si siano avviati correttamente con:

```
root:/dock/sophon# docker compose logs
```

[212] https://docs.docker.com/compose/
[213] https://docs.docker.com/engine/reference/commandline/images/

8.7 Configurazione del webserver dell'host

Si configuri il webserver dell'host per inoltrare tutto il traffico dalla porta 443 (o 80, se si è selezionato `http` in `DJANGO_PROXY_PROTOCOL`) alla porta locale 30033.

Sono allegate le istruzioni per il webserver Apache HTTPd[214]; possono essere però adattate se si vuole usare un webserver diverso, come NGINX[215] o caddy[216].

8.7.1 Con Apache HTTPd

Ci si assicuri che mod_rewrite[217], mod_proxy[218], mod_proxy_http[219] e mod_proxy_wstunnel[220] siano attivati.

Si aggiungano i seguenti `VirtualHost` alla configurazione:

```
<VirtualHost *:80>
    ServerName "ilmiosophon.it"
    ServerAlias "*.ilmiosophon.it"

    RewriteEngine On
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>
```

```
<VirtualHost *:443>
    ServerName "ilmiosophon.it"
    ServerAlias "*.ilmiosophon.it"

    SSLEngine on
    SSLCertificateFile      "/SOSTITUISCIMI/CON/IL/PERCORSO/ALLA/
↪FULL/CHAIN/SSL"
    SSLCertificateKeyFile   "/SOSTITUISCIMI/CON/IL/PERCORSO/ALLA/
↪CHIAVE/PRIVATA/SSL"

    ProxyPreserveHost On
    RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}

    RewriteEngine On
    RewriteCond %{HTTP:Upgrade} =websocket [NC]
    RewriteRule /(.*)           ws://127.0.0.1:30033/$1 [P,L]
    RewriteRule /(.*)           http://127.0.0.1:30033/$1 [P,L]

    Protocols h2 http/1.1
```

(continues on next page)

[214] https://httpd.apache.org/
[215] https://www.nginx.com/
[216] https://caddyserver.com/
[217] https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html
[218] https://httpd.apache.org/docs/2.4/mod/mod_proxy.html
[219] https://httpd.apache.org/docs/2.4/mod/mod_proxy_http.html
[220] https://httpd.apache.org/docs/2.4/mod/mod_proxy_wstunnel.html

(continua dalla pagina precedente)

```
    Header always set Strict-Transport-Security "max-age=63072000"
</VirtualHost>
```

Infine, si riavvii Apache HTTPd[221]:

```
root:/dock/sophon# systemctl restart httpd
```

8.8 Verificare il funzionamento

Se tutto è stato configurato correttamente, l'interfaccia web Sophon dovrebbe essere raggiungibile al dominio selezionato (`https://ilmiosophon.it`), e dovrebbe essere possibile effettuare il login con le credenziali configurate del primo *superutente*.

[221] https://httpd.apache.org/

Bibliografia

[matplotlib:histograms] Matplotlib. **Histograms**. https://web.archive.org/web/20210815080710/https://matplotlib.org/stable/gallery/statistics/hist.html

[jupyter:ifaq] Project Jupyter. **Institutional FAQ**. https://web.archive.org/web/20200317115530/https://jupyterhub.readthedocs.io/en/stable/getting-started/institutional-faq.html

[jupyter:hub] Project Jupyter. **JupyterHub**. https://web.archive.org/web/20211115163603/https://jupyterhub.readthedocs.io/en/stable/

[jupyter:kernels] Project Jupyter. **Jupyter kernels**. https://web.archive.org/web/20211118154819/https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

[jupyter:collaboration] Project Jupyter. **Real Time Collaboration**. https://web.archive.org/web/20211118174617/https://jupyterlab.readthedocs.io/en/stable/user/rtc.html

[wiki:eln] English Wikipedia. **Electronic lab notebook**. https://en.wikipedia.org/w/index.php?title=Electronic_lab_notebook&oldid=993314047

[overleaf:learn30mins] Overleaf. **Learn LaTeX in 30 minutes**. https://web.archive.org/web/20211116220924/https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

[so:survey2021] Stack Overflow. **2021 Developer Survey**. https://web.archive.org/web/20211126152610/https://insights.stackoverflow.com/survey/2021

[docker:overview] Docker. **Overview**. https://web.archive.org/web/20211121165850/https://docs.docker.com/get-started/overview/

[docker:networking] Docker. **Networking overview**. https://web.archive.org/web/20211111155419/https://docs.docker.com/network/

[docker:volumes] Docker. **Use volumes**. https://web.archive.org/web/20211121172749/https://docs.docker.com/storage/volumes/

[github:features] GitHub. **Features**. https://web.archive.org/web/20211124034005/https://github.com/features

Indice del modulo Python

HTTP Routing Table

/admin

/api

Indice analitico

D

E

F

T

U

V

W