CiteMatch

Cite with Confidence

Laboratorio di Architetture Software e Sicurezza Informatica

Ingegneria Informatica e Automatica

Università degli studi di Roma “La Sapienza”

a.a. 2023/2024

Sara Camussa -matricola 1982779

18/09/2024

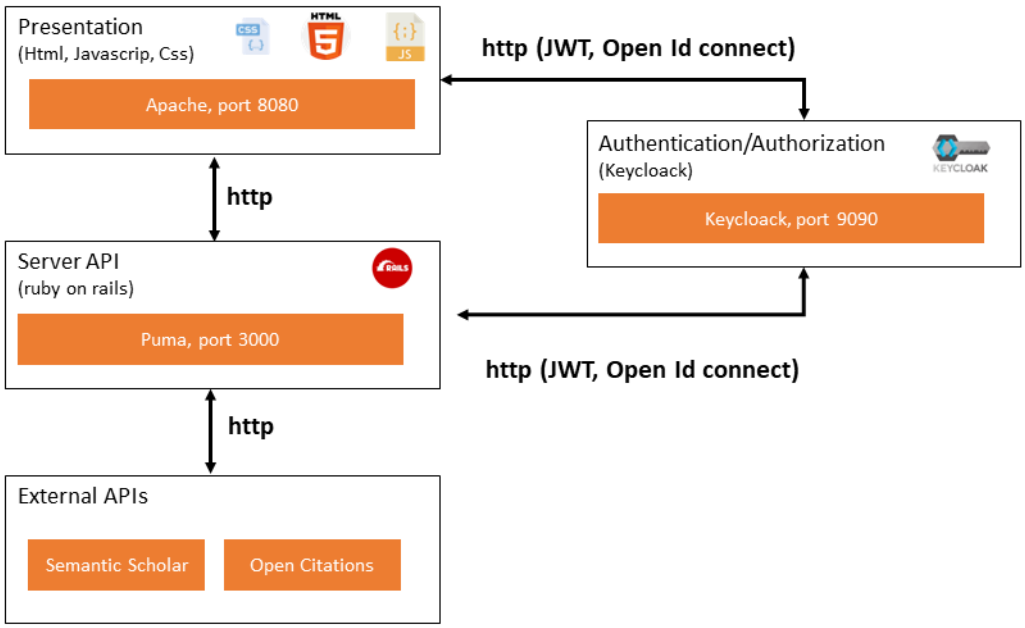
Sommario

1 INTRODUZIONE

CiteMatch è una Web Application che mette a confronto le citazioni di articoli scientifici presenti su due dei principali database pubblici oggi disponibili: **Semantic Scholar e Open Citations**. Partendo dalla selezione di un autore permette di risalire a tutte le sue pubblicazioni e citazioni presenti in Semantic Scholar. Una volta estratte le stesse informazioni presenti su Open Citations presenta all'utente una visione sintetica di tutte le citazioni evidenziando quelle presenti su uno solo dei due sistemi, quelle presenti su entrambi e quelle presenti solo su Open Citations che potrebbero essere aggiunte anche su Semantic Scholar in quanto la citazione è esiste su quest'ultimo come pubblicazione disponibile.

2 ARCHITETTURA DI RIFERIMENTO e TECNOLOGIE

L'architettura di riferimento scelta separa nettamente il front end dal back end dell'applicazione sia in termini di componenti software (Apache e Puma) che di tecnologie (Html,Javascript per il front end e Ruby on Rails per il backend). La gestione delle identità (IAM) viene effettuata tramite un componente centralizzato (Keycloack) che viene interfacciato mediante il protocollo standard Open Id Connect. Le API sviluppate interfacciano e coordinano le API esterne messe a disposizione da Semantic Scholar. Sia la componente di fronte-end che la componente di back-end vengono protette utilizzando il protocollo Open Id connect.



Vantaggi dell'architettura proposta

- Front End e Back End possono essere sviluppati indipendentemente (anche da persone distinte) e con tecnologie diverse. Il loro livello di accoppiamento è minimo e si basa esclusivamente sulla firma delle API REST.
- Front End e Back End possono scalare orizzontalmente in maniera del tutto indipendente a seconda dei carichi di lavoro a cui è sottoposta l'applicazione
- Front End e Back End possono evolvere indipendentemente e essere rilasciate separatamente se necessario
- Le API sviluppate e il Front End possono essere testati singolarmente facilitando la localizzazione e la risoluzione dei problemi.
- La gestione dell'autenticazione e dell'autorizzazione tramite Keycloack separa queste funzionalità dalla logica dell'applicazione, rendendo più semplice il mantenimento e l'evoluzione del sistema
- Gli utenti e i propri ruoli vengono gestiti in modo centralizzato e uniforme anche nell'ottica di implementare un sistema di SSO che gestisca più di una applicazione

3 USER STORIES

Scenario: Accesso al sistema e reindirizzamento alla pagina di login
Given L'utente vuole accedere al sistema e non è ancora autenticato.
When L'utente tenta di accedere al sistema
Then L'utente viene reindirizzato alla pagina di login

Scenario: Accesso al sistema con credenziali valide
Given L'utente è sulla pagina di login.
When L'utente inserisce username e password corrette
Then L'utente accede al sistema

Scenario: Accesso al sistema con credenziali non valide
Given L'utente è sulla pagina di login.
When L'utente inserisce username e password non valide
Then L'utente viene informato che le credenziali non sono corrette e l'accesso al sito viene bloccato

Scenario: Navigazione nell'applicazione tramite menu
Given L'utente è su una delle pagine dell'applicazione e vuole navigare sulla home, sulla pagina di ricerca o sui contatti
When L'utente seleziona la voce di menu
Then L'utente viene reindirizzato alla pagina selezionata

Scenario: Ricerca di un autore su Semantic Scholar
Given un autore con il nome "Jane Smith"
When l'utente inserisce il nome dell'autore nel campo di ricerca
Then il sistema estrae tutti gli autori su Scholar che corrispondono alla ricerca

Scenario: Navigazione tra le pagine di ricerca in avanti
Given L'utente visualizza la lista degli autori o la lista delle pubblicazioni
When l'utente preme il pulsante "next"
Then il sistema mostra la pagina di ricerca successiva

Scenario: Navigazione tra le pagine di ricerca indietro
Given L'utente visualizza la lista degli autori o la lista delle pubblicazioni
When l'utente preme il pulsante "previous"
Then il sistema mostra la pagina di ricerca precedente

Scenario: Selezione di un autore e visualizzazione delle sua pubblicazioni
Given l'utente ha effettuato la ricerca e gli viene presentata una lista di autori
When l'utente seleziona un autore
Then il sistema estrae tutte le pubblicazioni dell'autore selezionato e le mostra all'utente assieme a il dettaglio dell'utente selezionato

Scenario: Selezione di una pubblicazione e visualizzazione delle citazioni (pubblicazioni che citano quella selezionata)
Given l'utente visualizza la lista di pubblicazione dell'autore
When l'utente seleziona una pubblicazione
Then il sistema:

- Estrae tutte le citazioni da Scholar per l'autore e la pubblicazione selezionata
- Estra tutte le citazioni da Open Citations per l'autore e la pubblicazione selezionata
- Costruisce una lista unica delle citazioni evidenziando
    - Le citazioni che sono su Scholar ma non su Open Citation
    - Le citazioni che sono su entrambi i sistemi
    - Le citazioni che sono su Open Citations ma non in Scholar
- Per le citazioni presenti esclusivamente su Open Citations
    - Recupera i metadati mancanti della citazione (es. titolo, autori etc.)
    - Verifica se la citazione è presente su Scholar come articolo a se stante
    - In caso affermativo segnala che la citazione può essere aggiunta al database di Scholar

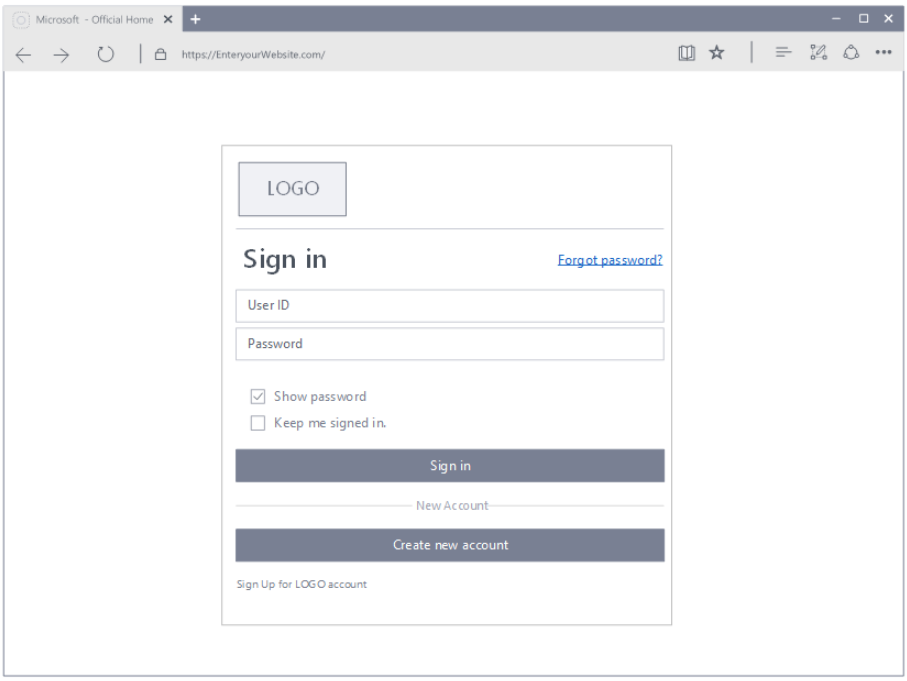
Scenario: Inserimento della citazione in Scholar con ruolo power user
Given l'utente ha visualizzato la lista delle citazioni inseribili
When l'utente seleziona l'azione di inserimento
Then il sistema "simula" l'inserimento su Scholar e mostra all'utente il risultato

Scenario: Inserimento della citazione in Scholar con ruolo normal user

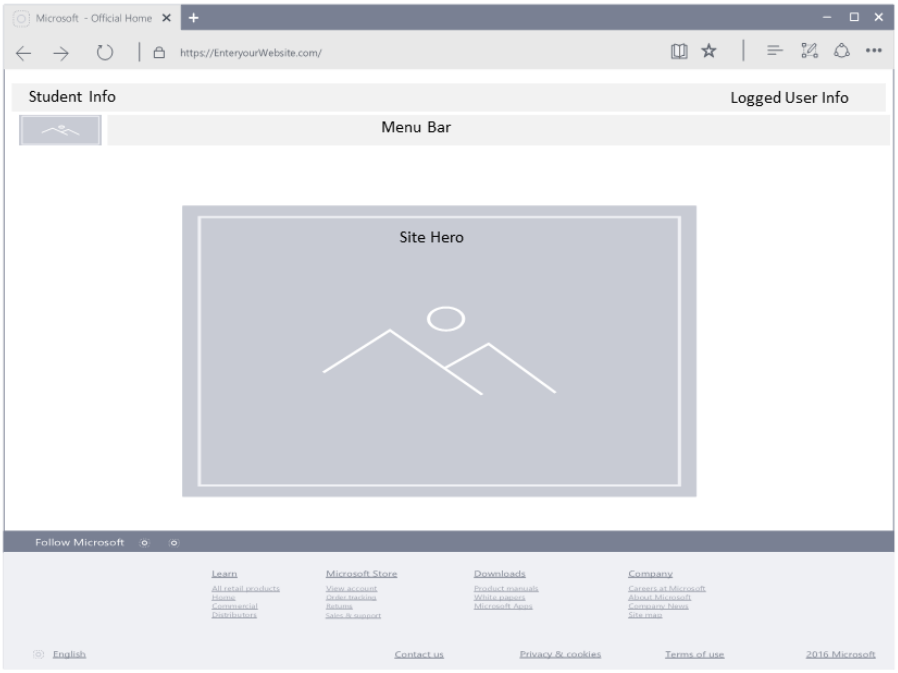
Given l'utente ha visualizzato la lista delle citazioni inseribili
When l'utente seleziona l'azione di inserimento
Then il sistema non permette la selezione dell'azione

4 MOCKUP

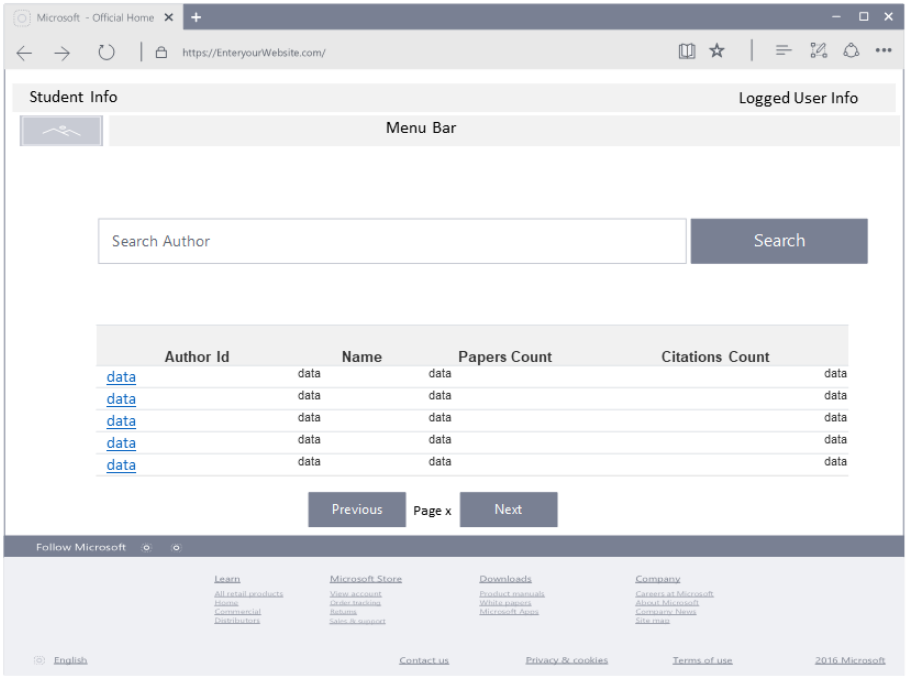
Login Page



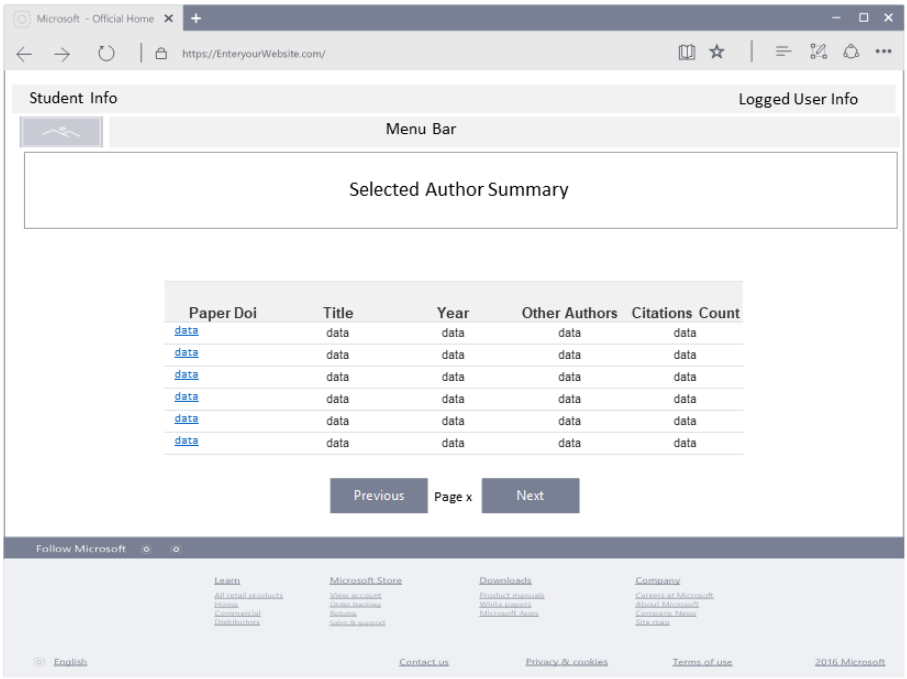
Home Page



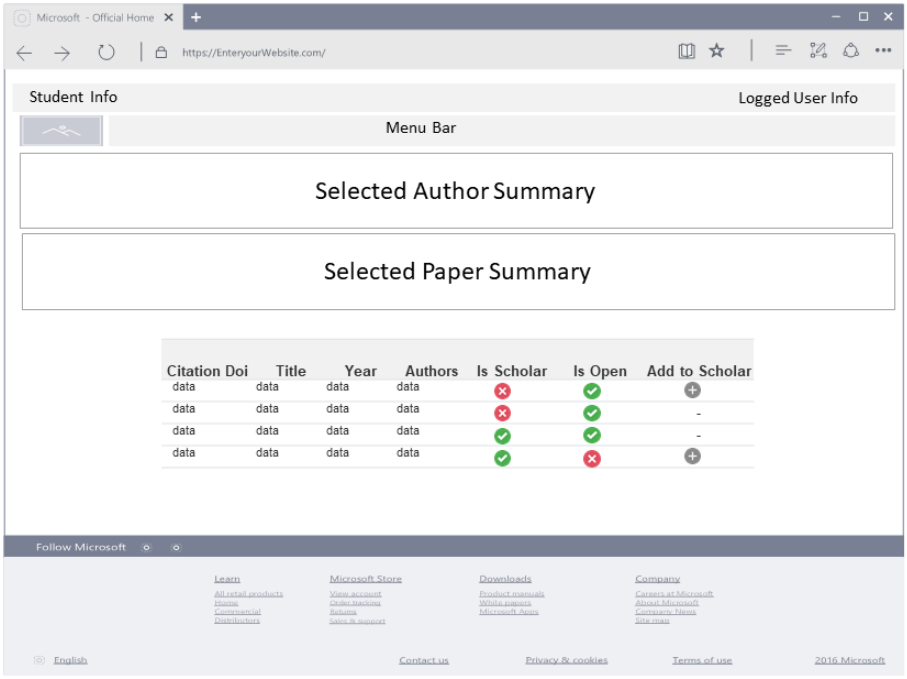
Search Author



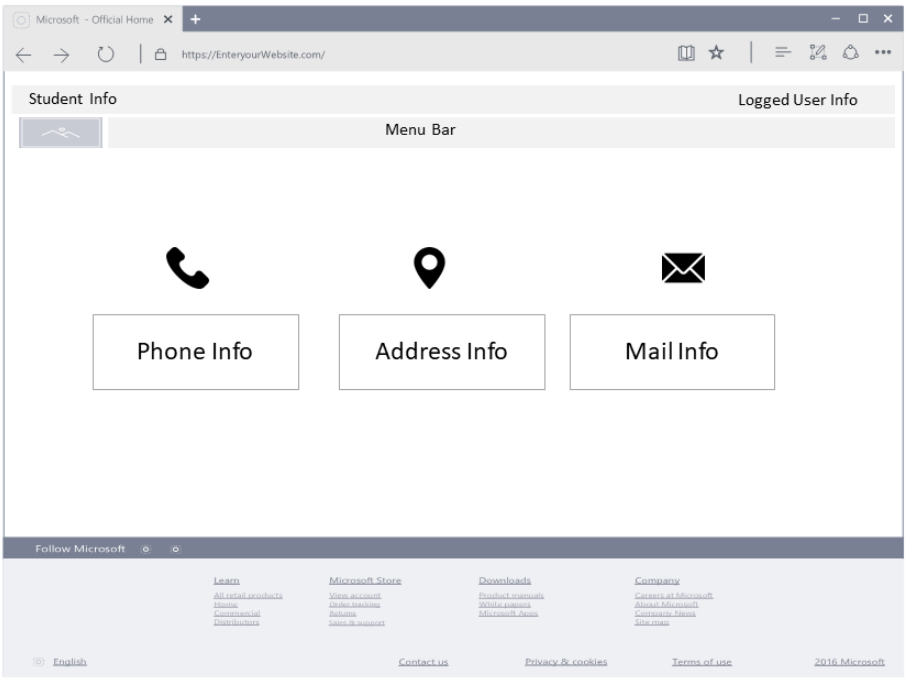
Publication List



Citation List

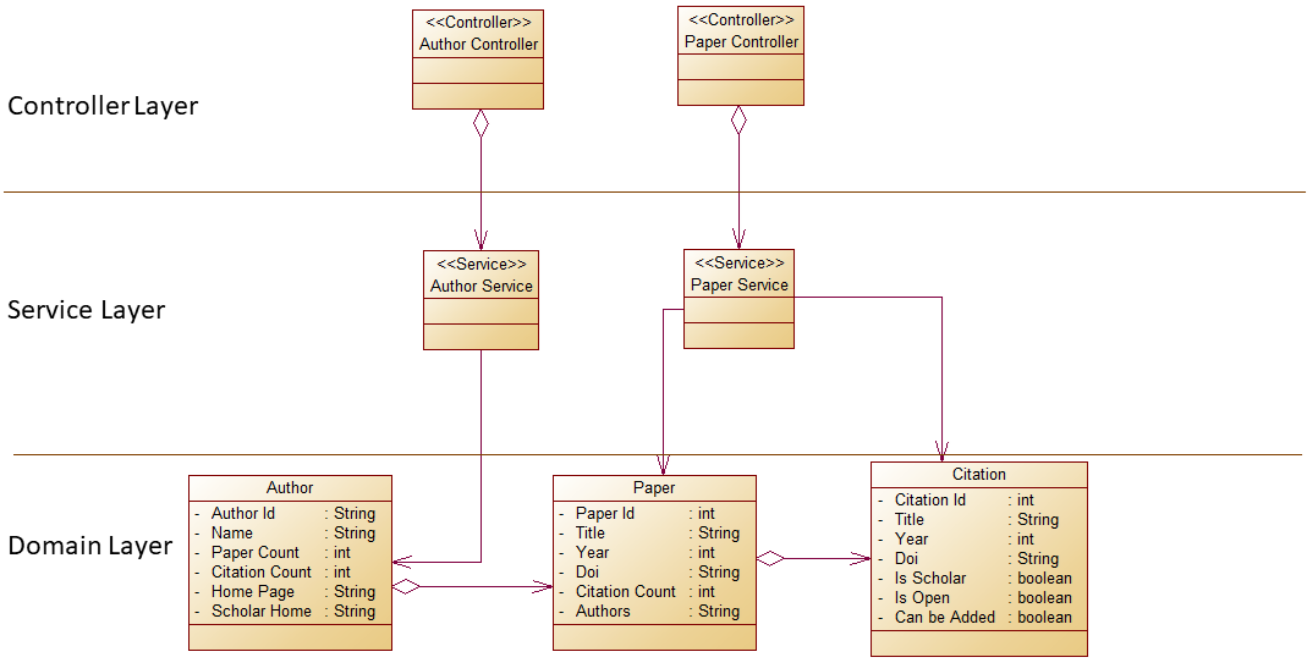


Contatti



Nel seguito viene mostrato un tipico esempio del flusso di navigazione tra le pagine. Resta inteso che la pagina di Login protegge tutte le pagine dell'applicazione e che la pagina dei Contatti è raggiungibile da qualsiasi altra pagina.

5 CLASS DIAGRAM



6 API INTERNE SVILUPPATE ED ESTERNE UTILIZZATE

L'applicazione di Back End espone una serie di API REST appositamente sviluppate che hanno il compito sia di disaccoppiare il front end dall'uso diretto delle API esterne, sia di implementare la logica di business necessaria alle funzionalità applicative. Tutte le API interne sono protette tramite keycloack per evitare accessi non autorizzati. La documentazione tramite swagger delle API sviluppate è eccedibile dal footer dell'applicazione (http://localhost:8080/citations/swagger/dist/index.html) .

API Sviluppate

URL	Metodo Utilizzo	Descrizione	Autenticazione
/author/search	GET	Ricerca full text di un autore	Si, tramite keycloack

/author/{author_id}/publication	GET	Lista delle pubblicazioni di un Autore	Si, tramite keycloack
/author/{author_id}	GET	Dettagli di un autore	Si, tramite keycloack
/paper/{paper_id}	GET	Dettagli di una pubblicazione	Si, tramite keycloack
/paper/{paper_id}/citations	GET	Estrae le citazioni sia da Scholar che da Open e effettua le operazioni di Merge	Si, tramite keycloack

Le API esterne utilizzate sono fornite da Semantic Scholar e Open Citations, sono API REST pubbliche e liberamente accessibili. Semantic Scholar richiede la registrazione e il conseguente rilascio di una API KEY necessaria ad innalzare il numero di richieste al secondo eseguibili sui loro sistemi.

API Esterne

URL	Metodo Utilizzo	Descrizione	Sorgente
/graph/v1/author/search	GET	Ricerca di un autore secondo vari parametri	Scholar
/graph/v1/author/{authorId}/papers	GET	Lista delle pubblicazioni di un Autore	Scholar
/graph/v1/author/{authorId}	GET	Dettagli di un Autore	Scholar
/graph/v1/paper/{paperId}/citations	GET	Estrae le citazioni di una pubblicazione	Scholar
/graph/v1/paper/{paperId}	GET	Dettagli di una pubblicazione	Scholar
/index/api/v2/citations/{paperId}	GET	Estrae le citazioni di una pubblicazione	Open Citations
/meta/api/v1/metadata/{paperId}	GET	Dettagli di una pubblicazione	Open Citations

7 CONTROLLO DEGLI ACCESSI

Il processo di Autenticazione e Autorizzazione avviene mediante il protocollo OpenID Connect e viene gestito tramite Keyclock (https://www.keycloak.org/), prodotto open source largamente utilizzato in ambito enterprise. La configurazione di Keycloack per l'applicativo ha comportato i seguenti passi:

- Creazione di un realm (spazio isolato per la gestione di utenti, ruoli e applicazioni) di nome **cit-realm**
- Creazione di un Client di tipo OpenID Connect di nome **cit-client**
- Creazione di 2 utenti cit-user-1 e cit-user-2
- Definizione di 2 ruoli cit-normal-user e cit-power-user assegnati rispettivamente a cit-user-1 e cit-user-2

Protezione del Front End

Il front end viene protetto utilizzando l'adapter Keycloack per Javascript in tutte le sue pagine secondo il seguente flusso:

- La pagina chiamata contatta il server Keycloack per verificare se l'utente è autenticato
- In caso negativo Keycloack reindirizza l'utente alla pagina standard di Login
- L'utente inserisce le credenziali
- Keycloak verifica le credenziali e, se sono corrette, rilascia all'applicazione un codice di autorizzazione
- L'applicazione scambia con Keycloack il codice di autorizzazione con un Access Token JWT che contiene le informazioni sull'utente e i suoi ruoli
- Il front end utilizza il token JWT per gestire gli aspetti di Autorizzazione alle singole funzionalità. Inoltre si occupa di inviare il token JWT alle API utilizzate in modo che quesse possano verificarne la validità e consentire le chiamate solo ad utenti precedentemente autenticati.

Protezione dei Back End

Tutte le API del Back end ricevono nell'Header http un token JWT generato da Keycloack. Per semplicità si è deciso di implementare il processo di autorizzazione in modalità **locale.**

In sostanza il Back End non necessità di contattare nuovamente Keycloack per verificare l'autenticità del token JWT ricevuto ma esegue questa verifica utilizzando un certificato memorizzato localmente contenente la chiave pubblica utilizzata da Keycloack per firmare il JWT e utilizzata a per verificare l'autenticità del JWT ricevuto.

Ruoli e Utenti disponibili

Utente	Ruolo	Funzionalità
cit-user-1	cit-normal-user	Accesso a tutte le pagine e funzionalità del sito. Non abilitato all'inserimento delle citazioni mancanti su Scholar
cit-user-2	cit-power-user	Accesso a tutte le pagine e funzionalità del sito. Abilitato all'inserimento delle citazioni mancanti su Scholar

Nice To Have

Nell'applicazione sviluppata, nuovi utenti e nuovi ruoli possono essere creati esclusivamente utilizzando il prodotto Keycloack. Nel caso si volesse implementare nell'applicazione un processo di registrazione di un nuovo utente, l'applicazione dovrebbe

(oltre ad implementare le opportune form di data entry) utilizzare le API messe a disposizione di Keycloack per gestire la creazione dell'utente e l'assegnazione degli opportuni ruoli.

8 TEST

Vista l'architettura dell'applicazione descritta nei paragrafi precedenti, i test sono stati effettuati utilizzando due diversi strumenti uno per il front end e uno per il back end.

Test del Front End

E' stato utilizzato lo strumento Cypress (https://www.cypress.io). Cypress è uno strumento di testing end-to-end (E2E), utilizzato principalmente per testare applicazioni web, offre una suite di strumenti per il testing automatico, semplificando il processo di scrittura, esecuzione e debugging di test per applicazioni JavaScript. Le User stories testate sono state le seguenti:

Scenario: Accesso al sistema e reindirizzamento alla pagina di login
Scenario: Accesso al sistema con credenziali valide
Scenario: Ricerca di un autore su Semantic Scholar
Scenario: Navigazione tra le pagine di ricerca in avanti
Scenario: Navigazione tra le pagine di ricerca indietro

I test effettuati possono essere reperiti all'interno della cartella citations\cypress\e2e del progetto Front End e sono i seguenti:

- **spec_login_1.cy**: Accesso all'applicativo tramite credenziali valide
- **spec_login_2.cy**: Accesso all'applicativo tramite credenziali valide e verifica dei ruoli assegnati all'utente
- **spec_author_search_1.cy**: Ricerca per autore e verifica del numero corretto di risultati ricevuti
- **spec_author_search_2.cy**: Ricerca per autore e verifica che i risultati ottenuti siano congruenti con i parametri di ricerca
- **spec_author_search_3.cy**: Ricerca per utente e verifica delle funzionalità di navigazione tra i risultati
- **spec_author_search_4.cy**: Ricerca per utente e verifica delle funzionalità di navigazione tra i risultati

Esempio di test Cypress

```
describe('Author Search', () => {
  it('should disable pagination buttons when search result returns less than 11 authors', () => {
    //Esegui il login su keycloack
    cy.origin('http://localhost:9090/', () => {
      cy.visit('http://localhost:8080/citations/search-author.html');
      cy.get('[id=username]').click().type('cit-user-1');
      cy.get('[id=password]').click().type('cit-user-1');
      cy.get('#kc-login').click();
    });
    cy.intercept('GET', 'http://localhost:3000/author/search*').as('getAuthorSearch');
    // Torna nel contesto dell'applicativo principale
    cy.origin('http://localhost:8080/', () => {
     // Imposta i parametri di ricerca
      cy.get('#author').type('camussa');
      cy.get('#fetchDataBtn').click();
     // Attendi che l'API sia terminata prima di proseguire
      cy.wait('@getAuthorSearch');

     // Verifica che il bottone "Previous" sia disabilitato
      cy.get('#prev-page').should('be.disabled');

     // Verifica che il bottone "Next" sia disabilitato
      cy.get('#next-page').should('be.disabled');
    });
  });
```

Test del Back End

I test del Back End sono stati realizzati tramite RSPEC. Le User stories testate sono state le seguenti:
Scenario: Accesso al sistema con credenziali valide
Scenario: Accesso al sistema con credenziali non valide
Scenario: Ricerca di un autore su Semantic Scholar
E' stato aggiunto un ulteriore test per verificare la validità delle regole di business di uno specifico oggetto di dominio (Author)

I test effettuati possono essere reperiti all'interno della cartella Citation\spec del progetto Back End e sono i seguenti:

- **authors_API_security.rb**: verifica che le API possono essere accedute solo se viene fornito un token JWT valido
- **authors_API_response.rb**: Ricerca per autore, verifica che la chiamata alla Api di ricerca di autore restituisca il numero di risultati corretti e che i risultati siano congruenti con i parametri di ricerca impostati
- **author_model.rb:** Verifica le regole di Business dell'oggetto di dominio Author

La particolarità dei test back end implementati è che devono gestire l'interazione con Keycloack per la generazione di un token JWT valido da utilizzare per la successiva chiamata alla API.

Esempio di test RSPEC

```ruby
require 'rails_helper'

RSpec.describe "Authors API", type: :request do
  describe "Test API Response" do

    let(:keycloak_url) { 'http://localhost:9090/realms/cit-realm/protocol/openid-connect/token' }
    let(:client_id) { 'cit-client' }
    let(:client_secret) { ENV['KEYCLOACK_CLIENT_SECRET'] }
    let(:username) { 'cit-user-1' }
    let(:password) { 'cit-user-1' }

    let(:token) do
      response = HTTParty.post(keycloak_url, body: {
        grant_type: 'password',
        client_id: client_id,
        client_secret: client_secret,
        username: username,
        password: password
      })
      JSON.parse(response.body)['access_token']
    rescue HTTParty::Error => e
      puts "HTTParty Error: #{e.message}"
      nil
    end

    let(:headers) { { "Authorization" => "Bearer #{token}" } }

    context "when the API is called" do
      it "It returns a valid response" do
        puts token
        get "/author/search", params: { query: "camussa" }, headers: headers
        expect(response).to have_http_status(:ok)
        json_response = JSON.parse(response.body)
        expect(json_response["Authors"]["total"]).to eq(5)
        expect(json_response["Authors"]["data"]).to be_an(Array)
        expect(json_response["Authors"]["data"].size).to eq(5)
        json_response["Authors"]["data"].each do |author|
            expect(author["name"]).to include("Camussa")
        end
      end
    end
  end
```