INDIGO IAM migration to Spring Authorization Server framework with a new customizable React user dashboard

Federica Agostini,[a] Alessandra Casale,[b] Jacopo Gasparetto,[a,*] Francesco Giacomini,[a] Davide Marcato,[c] Roberta Miccoli,[a] Enrico Vianello[a] and Stefano Zotti[a]

[a] *INFN-CNAF, Viale Berti Pichat 6/2, Bologna, Italy*
[b] *INFN-LNGS, Via Giovanni Acitelli 22, L'Aquila, Italy*
[c] *INFN-LNL, Viale dell'Università 2, Legnaro, Italy*
E-mail: federica.agostini@cnaf.infn.it, alessandra.casale@lngs.infn.it, jacopo.gasparetto@cnaf.infn.it, francesco.giacomini@cnaf.infn.it, davide.marcato@lnl.infn.it, roberta.miccoli@cnaf.infn.it, enrico.vianello@cnaf.infn.it, stefano.zotti@cnaf.infn.it

INDIGO Identity and Access Management (IAM) is a comprehensive service that enables organizations to manage and control access to their resources and systems effectively. It has been chosen as the AAI solution by the WLCG community and has been used since years by the INFN DataCloud, as well as by several other projects and experiments. INDIGO IAM is a Spring Boot application, based on OAuth/OpenID Connect technologies provided by the MITREid Connect library. A web interface based on the AngularJS framework is available to users and embedded into the INDIGO IAM service. The constant evolution of identity and access management systems like INDIGO IAM is imperative in the rapidly advancing landscape of cybersecurity and software development.

This contribution encapsulates the transformative journey of the INDIGO IAM software, transitioning from its existing above mentioned frameworks, to the more robust, secure, scalable and contemporary Spring Authorization Server with a React-based new dashboard.

International Symposium on Grids and Clouds (ISGC2024)
24 -29 March, 2024
Academia Sinica Computing Centre (ASGC), Institute of Physics, Academia Sinica Taipei, Taiwan

[*]Speaker

1. Introduction

Identity and Access Management (IAM) is a crucial component of modern IT infrastructures, enabling organizations to securely manage and control access to their systems and data. INDIGO IAM [1] is a powerful IAM service designed to meet the complex and evolving needs of research and academic institutions served by INFN and by other computing centers distributed in Europe. Developed as a part of the European Commission-funded INDIGO DataCloud project [2], INDIGO IAM provides a comprehensive solution for managing identities, roles, and access policies within a distributed and heterogeneous environment. The IAM service has been selected by the WLCG management board to be the core of the future, token-based WLCG AAI and exemplifies INFN commitment for the foreseeable future, with the current support of several Italian and European projects ([3], [4], [5], [6]).

The ongoing development of identity and access management systems, such as INDIGO IAM, is crucial in the ever-changing field of cybersecurity and software development. This article outlines the evolution of the INDIGO IAM software, moving away from its previous frameworks, MITREid Connect [8] and AngularJS [9] web interface, to the modern Spring Authorization Server [10] with a new dashboard based on React [11].

2. INDIGO IAM

INDIGO IAM is a Spring Boot application, based on OAuth/OpenID Connect technologies ([12], [13]) and the MITREid Connect library. It offers support for multiple authentication mechanisms, such as local authentication, SAML identities and federations like EduGAIN [14], social identity providers such as Google [15] or GitHub [16], etc. INDIGO IAM allows a user to link several authentication credentials (OpenID Connect and SAML accounts, but also X.509 certificates and SSH keys) to a single account. The platform provides a registration service for both moderated and automatic user enrollment, with the option to disable it as needed, and supports the enforcement of Acceptable Use Policies (AUP). INDIGO IAM exposes identity information, attributes and capabilities to services through JWT tokens [17], including the capability of delegation and token renewal. Thanks to its integration with OpenID Connect and OAuth, the system seamlessly integrates with ready-to-use components. A Virtual Organization Membership Service Attribute Authority (VOMS AA) microservice replaces the legacy VOMS [18] functionalities. In particular, it allows for Virtual Organization access control in distributed services and can easily integrate with existing VOMS-aware services. Figure 1 shows an high level overview of the INDIGO IAM architecture.

An interactive web interface, utilizing the AngularJS framework, is embedded into the INDIGO IAM service, providing users with easy accessibility. Within this interface, users can navigate various features and functionalities. Figure 2 illustrates a snapshot of the IAM dashboard, showcasing different aspects. On the right side, the homepage is displayed, tailored for administrator users, offering a comprehensive overview. Meanwhile, on the left side, views of other pages (i.e. the list of OAuth Client registered in INDIGO IAM and the available System Scopes) are presented, accessible by clicking on their respective tabs.

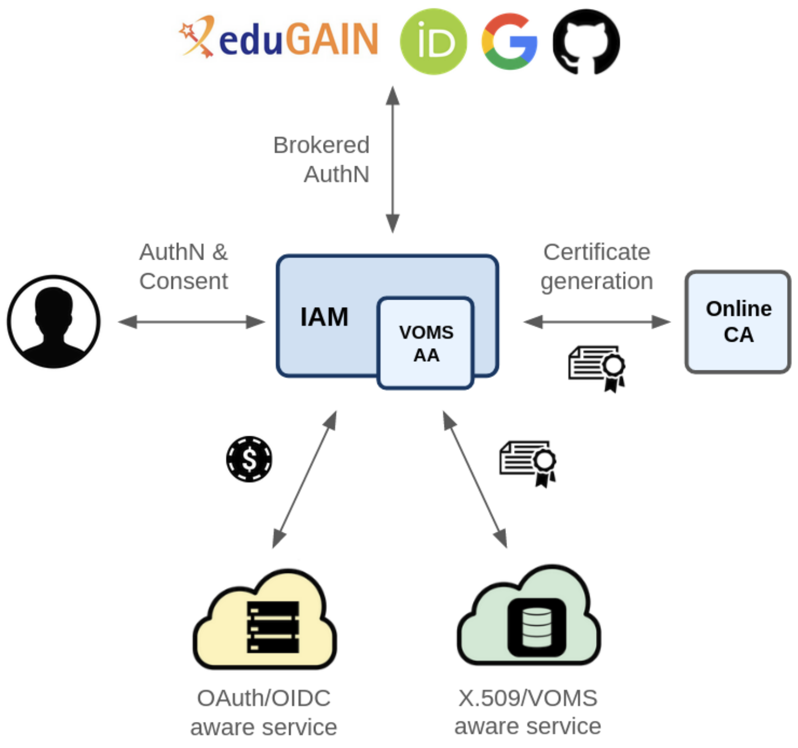


Figure 1: The INDIGO IAM overview, including the VOMS AA microservice. IAM is responsible for user and agent authentication, supporting several authentication mechanisms and exposing identity information through standard OpenID Connect interfaces. This approach simplifies the integration in relying services.

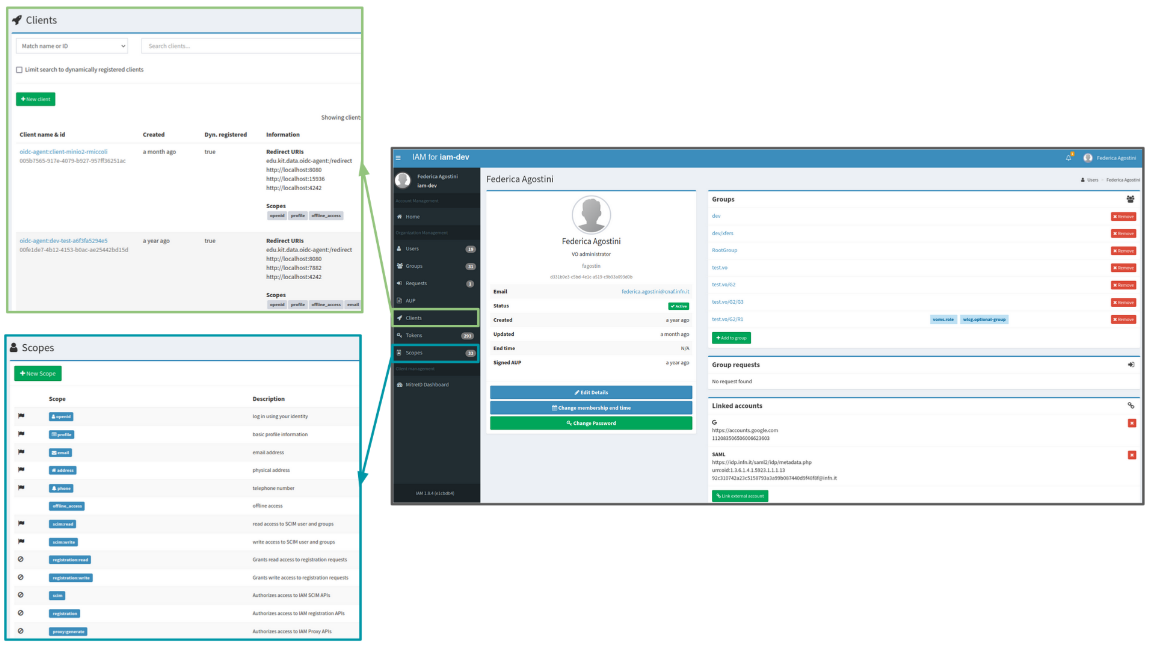
2.1 INDIGO IAM deployment

The INDIGO IAM service is structured to optimize performance and scalability. It usually operates behind an NGINX reverse proxy, ensuring efficient handling of incoming requests. Data management is facilitated by a MySQL database, providing a reliable storage solution. Horizontal scalability is achieved through the utilization of Redis, which efficiently manages sessions and external caching.

Deployed as a containerized service on Kubernetes, INDIGO IAM benefits from advanced orchestration capabilities. Autoscaling ensures that resources are dynamically allocated based on demand, allowing for seamless handling of fluctuating workloads. Moreover, zero downtime rolling updates guarantee continuous availability and uninterrupted service delivery, enhancing overall reliability and user experience. To date, around 30 instance of INDIGO IAM are deployed at the INFN-CNAF for experiments and project of different purposes, while about 10 IAM services are deployed outside CNAF with custom configuration.

2.2 Current development

The current development roadmap of INDIGO IAM is centered in elevating the functionality, efficiency, and security of the platform. At the forefront of these endeavors is a comprehensive



Clients

Match name or ID | Search clients...

☐ Limit search to dynamically registered clients

+ New client

Showing client

Client name & id	Created	Dyn. registered	Information
oidc-agent:client-minio2-rmiccoli 005b7565-917e-4079-b927-957ff36251ac	a month ago	true	**Redirect URIs** edu.kit.data.oidc-agent:/redirect http://localhost:8080 http://localhost:15936 http://localhost:4242 **Scopes** openid profile offline_access
oidc-agent:dev-test-a6f3fa5294e5 00fe1de7-4b12-4153-b0ac-ae25442bd15d	a year ago	true	**Redirect URIs** edu.kit.data.oidc-agent:/redirect http://localhost:8080 http://localhost:7882 http://localhost:4242 **Scopes** openid profile offline_access email

Scopes

+ New Scope

	Scope	Description
⚑	openid	log in using your identity
⚑	profile	basic profile information
⚑	email	email address
⚑	address	physical address
⚑	phone	telephone number
	offline_access	offline access
⚑	scim:read	read access to SCIM user and groups
⚑	scim:write	write access to SCIM user and groups
⊘	registration:read	Grants read access to registration requests
⊘	registration:write	Grants write access to registration requests
⊘	scim	Authorizes access to IAM SCIM APIs
⊘	registration	Authorizes access to IAM registration APIs
⊘	proxy:generate	Authorizes access to IAM Proxy APIs

IAM for **iam-dev** Federica Agostini

Federica Agostini iam-dev

Account Management
Home
Organization Management
Users 19
Groups 31
Requests 1
AUP
Clients
Tokens 293
Scopes 33
Client management
MitreID Dashboard

IAM 1.8.4 (e1cbdb4)

Federica Agostini

Users > Federica Agostini

Federica Agostini
VO administrator
fagostin
d331b9e3-c5bd-4e1c-a519-c9b93a093d0b

Email	federica.agostini@cnaf.infn.it
Status	✓ Active
Created	a year ago
Updated	a month ago
End time	N/A
Signed AUP	a year ago

Edit Details
Change membership end time
Change Password

Groups

dev		Remove
dev/xfers		Remove
RootGroup		Remove
test.vo		Remove
test.vo/G2		Remove
test.vo/G2/G3		Remove
test.vo/G2/R1	voms.role wlcg.optional-group	Remove

+ Add to group

Group requests

No request found

Linked accounts

G
https://accounts.google.com
112083506506006623603

SAML
https://idp.infn.it/saml2/idp/metadata.php
urn:oid:1.3.6.1.4.1.5923.1.1.1.13
92c310742a23c5158793a3a99b087440d9f48f8f@infn.it

Link external account

Figure 2: Screenshots of the current INDIGO IAM dashboard based on AngularJS. Right side: homepage displayed to an administrator user. Left side: views of other pages (list of OAuth Clients and System Scopes) as example, accessible by clicking on their respective tabs.

overhaul of our auditing capabilities, with a focus on enhancing tracking, analysis, and reporting functionalities. This initiative will empower users and administrators alike with deeper insights into system activities and interactions. Another key area of the roadmap, main topic of this article, is addressing obsolete dependencies to ensure the longevity and relevance of our technology stack. This involves transitioning from legacy frameworks such as MITREid to more modern and robust solutions like Spring Authorization Server, as well as migrating from AngularJS to the increasingly popular and versatile React JS framework. Among others, an advantage of this transitioning is to streamline development processes, future-proof the service against evolving industry standards, etc.

Moreover, the efforts are directed towards improving usability for both users and administrators, enhancing scalability, and boosting performance. Specifically, the measures implemented to reach the goal include not storing access tokens in the database, introducing a dedicated garbage collector service to remove unused data, implementing fine-grained authorization with an external, more reliable service such as Open Policy Agent [19], etc. In addition, interoperability is a central objective, with initiatives to support OIDC Federations [20] and improve conformance with AARC Blueprint Architecture [21] and its associated guidelines.

Finally, security remains the central point of the service, and the focus is for continuously enhancing the defenses to safeguard against evolving threats and vulnerabilities. One notable initiative in this domain is the integration of Multi-Factor Authentication, which adds an additional layer of protection to user accounts and sensitive data.

3. Migration to Spring Authorization Server

The MITREid framework has served as a reliable foundation for INDIGO IAM, providing essential identity management capabilities. However, in order to address the requirements for improved security, scalability and modern features, the decision was made to transition to Spring Authorization Server. This framework, built on top of Spring Security, provides a secure, lightweight and customizable foundation for building an OAuth 2.1 and OpenID Connect 1.0 Authorization Server implementation thus offering a more robust and flexible IAM solution. This migration signals a strategic effort to bring INDIGO IAM up to date with the latest industry standards and practices. Spring Authorization Server boasts a wide range of features, including support for OAuth 2.1 draft and OpenID Connect, adaptive authentication, and a modular architecture that enables seamless integration with other Spring ecosystem components.

The forked and self-maintained version of the MITREid Connect library that INDIGO IAM currently relies on has not seen any significant support or evolution in years. Transitioning to Spring Authorization Server allows for alignment with the natural evolution of the current Java/Spring-based architecture. This change also ensures long-term support and easier maintainability, as well as better compliance with OIDC/OAuth standards.

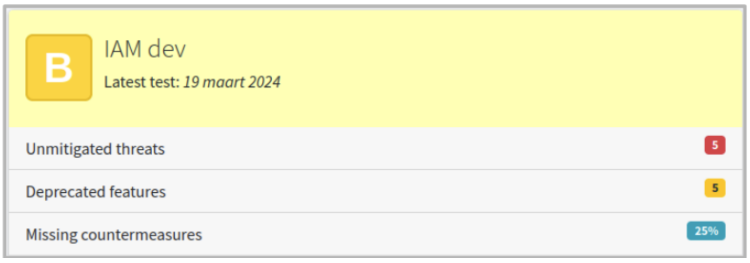
3.1 Proof of concept

In order to showcase the capabilities of the Spring Authorization Server, preliminary tests were conducted using the OAuch [22] tool, a compliance testing framework for the OAuth 2.0 protocol. OAuch thoroughly evaluates the adherence to standards and mitigation of known threats in the implementation of an OAuth 2.0 Authorization Server. The tool generates a detailed report based on the test analysis and identifies any potential vulnerabilities.

The comparison was made between a CNAF development instance of IAM based on the MITREid Connect library and the OAuth 2.0 standard, and a rough application as proof of concept built on top of Spring Authorization Server. The results (Figure 3 and 4) confirmed what has been said theoretically so far, namely that switching to Spring Authorization Server increases compliance and support for OAuth/OpenID Connect standards. More specifically, it was verified that Spring Authorization Server already supports many standard OAuth grant types (also known as flows) and that many OIDC/OAuth endpoints are supported by default without requiring any additional development. Tests on the entire set of functionalities currently supported by INDIGO IAM are still in progress.

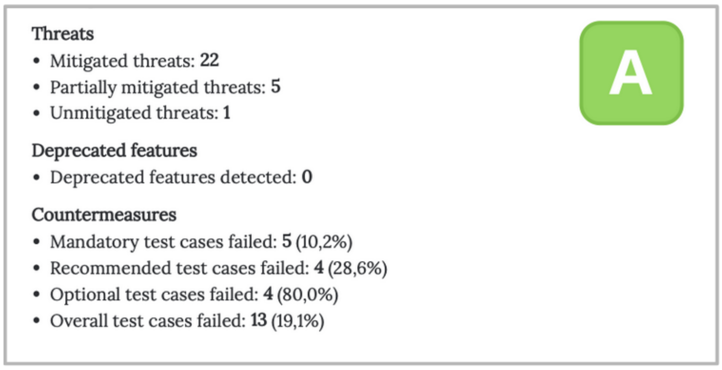
4. A React based new dashboard

Currently, INDIGO IAM provides the OIDC/OAuth functionalities, the REST API endpoints and a custom web dashboard based on the MITREid library, extended with custom AngularJS components, as one single service. Since both MITREid and AngularJS are now discontinued, the need for a new modern dashboard naturally arises. On the other hand, given this necessity, to improve the current user interface and user experience embracing modern web design techniques becomes an opportunity, as well as making the development and maintenance processes more sustainable using tools and frameworks extensively adopted by the web development community. Another key


B
IAM dev
Latest test: *19 maart 2024*

Unmitigated threats	5
Deprecated features	5
Missing countermeasures	25%

Figure 3: Result of OAuch test against the current IAM instance based on MITREid Connect


Threats
- Mitigated threats: **22**
- Partially mitigated threats: **5**
- Unmitigated threats: **1**

Deprecated features
- Deprecated features detected: **0**

Countermeasures
- Mandatory test cases failed: **5** (10,2%)
- Recommended test cases failed: **4** (28,6%)
- Optional test cases failed: **4** (80,0%)
- Overall test cases failed: **13** (19,1%)

A

Figure 4: Result of OAuch test against the proof of concept based on Spring Authorization Server

aspect for the development of the new dashboard is the complete decoupling of its source code from the INDIGO IAM code-base. In this new context, the INDIGO IAM core service, based on Spring Authorization Server will be responsible, together with the above mentioned tasks, for the authentication and authorization procedures as well as for serving the APIs, while the dashboard will be a completely independent service. With this model, the core service can be deployed headless without any dashboard if not required, therefore making the deployment more flexible. Because of the MITREid dashboard requires a Jakarta Server Pages (JSP) session to interact with the API endpoints, the INDIGO IAM APIs currently must support also this authentication method other than OIDC/OAuth, increasing the complexity and reducing the maintainability.

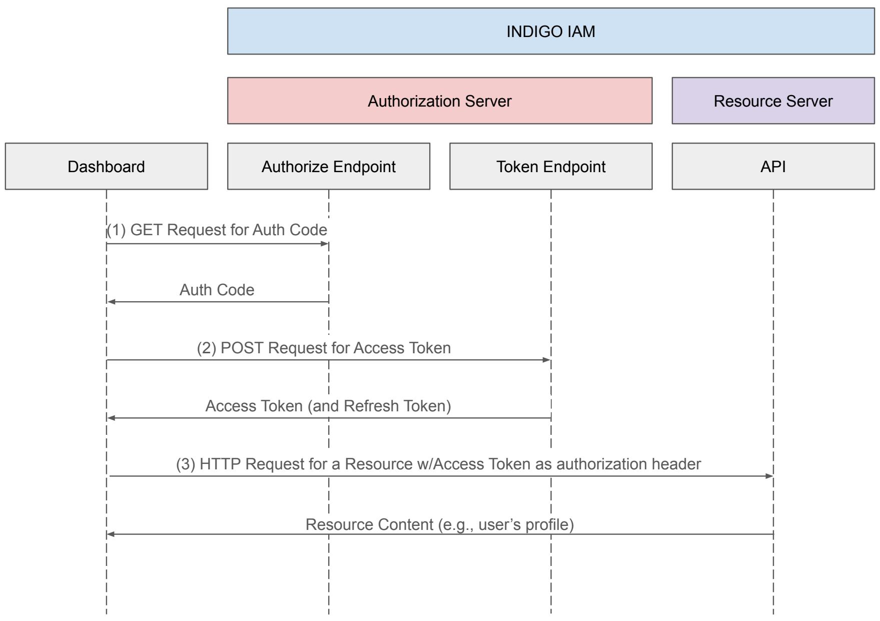

INDIGO IAM
Authorization Server
Resource Server
Dashboard
Authorize Endpoint
Token Endpoint
API
(1) GET Request for Auth Code
Auth Code
(2) POST Request for Access Token
Access Token (and Refresh Token)
(3) HTTP Request for a Resource w/Access Token as authorization header
Resource Content (e.g., user's profile)

Figure 5: Authorization Code flow (PKCE not shown). 1. The user clicks on the login button and they are redirected to the Authorize Endpoint. After a successful log in, they are redirected back to the dashboard with the Authorization Code. 2. The dashboard makes a POST request to the Token Endpoint, providing the previously received Authorization Code. If the user is authorized to receive the claims they asked for, they receive an Access Token (and optionally a Refresh Token). 3. The dashboard can now query the INDIGO IAM APIs with the received Access Token.

Using the OAuth terminology, the new IAM core service will play both the roles of the Authorization Server for the OIDC/OAuth procedures, and the Resource Server for what concerns the APIs, as shown in Figure 5. Since INDIGO IAM naturally supports OIDC/OAuth, its usage for the dashboard login would be ideal, making the dashboard a full-fledged IAM Client which processes protected resources (the IAM endpoints) on behalf of the user. As a consequence, the JSP session to access the API is not required anymore and can be removed simplifying the code and making it less prone to bugs and vulnerabilities.

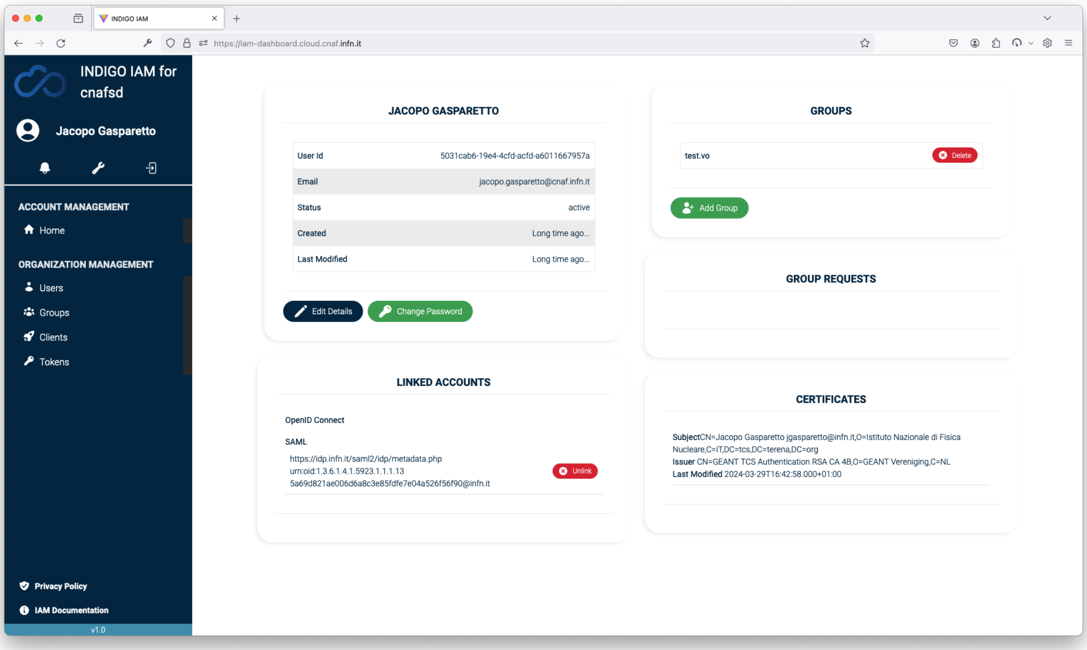
The choice of the technological stack fell onto the popular HTML 5, JavaScript/TypeScript and CSS web stack, that is the de facto state of the art of web development, making it the most reasonable option. On the other hand, React has been chosen as rendering framework for its simple declarative and component-based architecture, which offers a modern and efficient approach for building interactive and responsive user interfaces and web applications. React is also widely used by a huge number of major players and supported by a large community, an aspect that make the development more sustainable facilitating the inclusion of new collaborators.

The adoption of React also yields the advantage of creating reusable components for subsequent web applications developed by INFN. By leveraging the React component-based architecture, developers can encapsulate specific functionalities into modular units that can be easily reused across different projects. This not only streamlines the development process but also enhances maintainability and scalability, as updates or modifications to these components can be propagated seamlessly throughout the application ecosystem.

4.1 Proof of Concept

To assess the feasibility of the previously described requirements, a preliminary test dashboard has been developed as a full browser-based Single Page Application (SPA) which runs exclusively on the browser. The OIDC/OAuth responsibilities also run within the browser. In particular, the Authorization Code grant [12] with the Proof for Key Code Exchange (PKCE) [23] OAuth extension is used by the web application (the Client) to authenticate and authorize the user, receiving an Access Token from the INDIGO IAM service (Authorization Server) that the dashboard will then use to make HTTP requests to the INDIGO IAM protected API endpoints (Resource Server), as shown in Figure 5.

An example of the new INDIGO IAM dashboard based on the React framework is shown in Figure 6. At the time of writing, the implementation includes the homepage of an IAM administrator or normal user. Left tab should still incorporate the functionalities shown in the current dashboard (Figure 2).



User Id	5031cab6-19e4-4cfd-acfd-a6011667957a
Email	jacopo.gasparetto@cnaf.infn.it
Status	active
Created	Long time ago...
Last Modified	Long time ago...

test.vo

OpenID Connect

SAML

https://idp.infn.it/saml2/idp/metadata.php
urn:oid:1.3.6.1.4.1.5923.1.1.1.13
5a69d821ae006d6a8c3e85fdfe7e04a526f56f90@infn.it

SubjectCN=Jacopo Gasparetto jgasparetto@infn.it,O=Istituto Nazionale di Fisica Nucleare,C=IT,DC=tcs,DC=terena,DC=org
Issuer CN=GEANT TCS Authentication RSA CA 4B,O=GEANT Vereniging,C=NL
Last Modified 2024-03-29T16:42:58.000+01:00

Figure 6: Example of the new React-based INDIGO IAM homepage as Single Page Application.

4.2 Security Concerns

Given the sensitive nature of INDIGO IAM, prioritizing security is imperative and every security flow must be considered. On one hand, the SPA model, which is completely browser-based, makes development simple and direct, allowing for a quick and efficient implementation of a lightweight and high-performance application. On the other hand, it presents security issues. Due to its nature, the static website cannot be protected by confidential secrets and must therefore be a *public* IAM Client. As a result, it becomes challenging to prevent an unauthorized user from requesting more privileges than they are allowed to get, since the same OAuth Client constantly shares the same scopes and claims for all users. To overcome this limitation, a custom implementation of INDIGO IAM APIs would be required to perform the endpoint authorization on the basis of user identity. Additionally, a more significant issue is due to the exposure of the Access Token and Refresh Token to the JavaScript code. If not properly mitigated with PKCE extension, which is mandatory for a full browser-based app, and Cross Site Request Forgery countermeasures to contrast phishing, a user can be induced to land on a maliciously crafted website that, after a legitimate Authorization Code flow, can steal the Access Token. When the access token is in the hands of the attacker, they can make legitimate requests to the Resource Server on behalf of the user. Even worse, if the attacker is able to collect the Refresh Token, once the Access Token expires they can ask for a new set of credentials without the user intervention.

4.3 Exploring a backend mediated dashboard implementation

In order to overcome these issues, a backend can be inserted between the dashboard and the INDIGO IAM core service as security layer to offload the OAuth responsibilities from the browser to the server. It is possible to identify two main backend designs: a Mediating-Token Backend (MTB) and a full Backend For Frontend (BFF). The MTB is essentially a small service that processes the Authorization Code flow server side on behalf of the user and returns the Access Token to the dashboard, establishing with it a cookie-based session to identify the user. The dashboard can thus directly perform requests to the Resource Server with the Access Token until it expires. Once the Access Token is expired, the dashboard can resume the cookie-based session and asks the backend to perform a Refresh Flow and obtain a new Access Token. This architecture mitigates the risk of Refresh Token exploitation, since no Refresh Token is exposed to the JavaScript code. Moreover, it permits to secure the OAuth Client with a confidential secret and allows for custom fine-grained authorization mechanism. Nevertheless, this is considered only a partial solution. This architecture is slightly more secure than a full browser-based web application, but since the JavaScript code still possesses the Access Token and makes requests to the Resource Server with it, it shares some of the same attack surface.

The BFF architecture is an extension of the MTB that, beyond the OAuth responsibilities, proxies every request from the dashboard to the Resource Server. In this framework, the web Client receives no tokens at all and thus there are no risks of token exploitation. Once the Authorization Code flow is completed server side, a cookie-based session between the web application and the BFF is established, and every request made by the frontend is proxied by the backend, which augments them with the Access Token associated to the user session. Of course, this complexity increases implementation challenges but provides enhanced security by avoiding exposure of any confidential information to the public JavaScript code.

4.3.1 Next.js

To develop a backend component as described above, the *Next.js* [24] framework is currently under exploration. In addition to being the officially recommended framework by React developers, it offers many solutions to build server-side rending web applications, exploiting the same programming paradigms used to build a simple static SPA based on React, making relatively easy the porting from a client-side rendering website to server-side rendering based application. A preliminary experiment of OIDC/OAuth server-side authentication and authorization, using the *NextAuth.js* [25] library for Next.js, was successful, demonstrating the feasibility of this approach. Next.js also offers out of the box tools to implement cookie-based sessions, which, in combination with the server-side authentication and authorization mechanisms, makes it an ideal candidate to build a full BFF to increase the overall security of the INDIGO IAM dashboard.

5. Future looks and conclusions

In conclusion, the future of INDIGO IAM looks promising with the migration to Spring Authorization Server and the development of a new dashboard. These steps will not only improve compliance with standards but also ensure a more supported and modern framework. The transition to a Single-Page App built in React has shown great potential for enhancing user interface and security. Overall, these developments will further solidify INDIGO IAM as a critical service for scientific communities.

Acknowledgements

The work presented in this paper has been supported by the NextGenerationEU European initiative through the Italian Ministry of University and Research, PNRR Mission 4, Component 2 - ICSC [5]: Investment 1.4, Project code CN00000013 - CUP I53C21000340006; TeRABIT [6]: Investment 3.1, Project code IR0000022 - CUP I53C21000370006. This research was also co-funded by the Italian Complementary National Plan PNC-I.1 "Research initiatives for innovative technologies and pathways in the health and welfare sector" D.D. 931 of 06/06/2022, "DARE [7] - DigitAl lifelong pRevEntion" initiative, code PNC0000002.

References

[1] *INDIGO IAM*, URL https://indigo-iam.github.io/v/current/

[2] *INDIGO DataCloud*, URL https://www.indigo-datacloud.eu/, last seen April 2024

[3] *EOSC beyond*, URL https://eosc.eu/eu-project/eosc-beyond/, last seen April 2024

[4] *AARC*, URL https://aarc-project.eu/, last seen April 2024

[5] *ICSC*, URL https://www.supercomputing-icsc.it/en/icsc-home/, last seen April 2024

[6] *TeRABIT*, URL https://www.terabit-project.it/it/, last seen April 2024

[7] *DARE*, URL https://www.fondazionedare.it/en/, last seen April 2024

[8] *MITREid Connect*, URL https://github.com/mitreid-connect/

[9] *AngularJS — Superheroic JavaScript MVW Framework*, URL https://angularjs.org/, last seen April 2024

[10] *Spring Authorization Server*, URL https://spring.io/projects/spring-authorization-server

[11] *React*, URL https://react.dev/, last seen April 2024

[12] *The OAuth 2.0 Authorization Framework* , DOI 10.17487/RFC6749, URL https://www.rfc-editor.org/rfc/rfc6749

[13] *OpenID Connect Core 1.0*, URL https://openid.net/specs/openid-connect-core-1_0.html

[14] *EduGAIN interfederation*, URL http://www.geant.org/Services/Trust_identity_and_security/eduGAIN, last seen April 2024

[15] *The Google Identity Platform*, URL https://developers.google.com/identity, last seen April 2024

[16] *The Github OAuth API reference*, URL https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps, last seen April 2024

[17] *JSON Web Tokens*, DOI 10.17487/RFC7519, URL https://datatracker.ietf.org/doc/html/rfc7519

[18] *Virtual Organisation Membership Service (VOMS)*, DOI 10.5281/zenodo.12634651, URL https://italiangrid.github.io/voms

[19] *Open Policy Agent*, URL https://www.openpolicyagent.org/, last seen April 2024

[20] *OpenID Federation 1.0*, URL https://openid.net/specs/openid-federation-1_0.html, last seen April 2024

[21] *AARC Blueprint Architecture*, URL https://aarc-community.org/architecture/, last seen April 2024

[22] *OAuch*, URL https://oauch.io/, last seen April 2024

[23] *Proof for Key Code Exchange*, DOI 10.17487/RFC7636, URL https://www.rfc-editor.org/rfc/rfc7636

[24] *Next.js*, https://nextjs.org, last seen April 2024

[25] *NextAuth*, https://next-auth.js.org, last seen April 2024