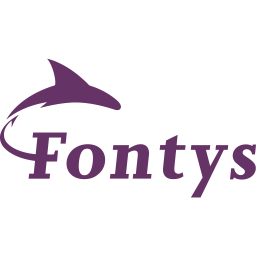
CIRI2 Auth0 Document

Jordy Walraven



**Version history:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author(s) | Description | Date |
| 1 | Jordy Walraven | Setup Initial Auth0 documentation | 14-4-2024 |

**Distribution history:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Distributed to | Comments | Date |
|  |  |  |  |

Table of contents

[Context 2](#_Toc163995457)

[Application architectural styles 4](#_Toc163995458)

[C4 Architecture 4](#_Toc163995459)

[Level1 4](#_Toc163995460)

[Level 2 6](#_Toc163995461)

[**Actors and Their Interactions** 6](#_Toc163995462)

[**PC Gamer** 6](#_Toc163995463)

[**Admin** 6](#_Toc163995464)

[**Containers and Their Interactions** 7](#_Toc163995465)

[**Single Page Application** 7](#_Toc163995466)

[**API Gateway** 7](#_Toc163995467)

[**Microservices** 7](#_Toc163995468)

[**Databases** 7](#_Toc163995469)

[**Diagram Description** 8](#_Toc163995470)

# Context

Ciri2 is a platform designed for PC gaming enthusiasts. At its core, Ciri2 serves as a hub where users can assess their system's capability to run specific games and benchmark their performance against others in the community. Here's an overview of its key functionalities:

**Account Creation:**

Users can easily create personalized accounts, unlocking access to a host of features tailored to their gaming needs.

**PC Rig Configuration:**

Within the application, users have the flexibility to construct and customize their PC rigs using intuitive part pickers for various components. This empowers them to tailor their systems to meet the demands of their favorite games.

**Game Overview:**

Through Ciri2, users gain access to a comprehensive overview of games, including key details such as system requirements, gameplay mechanics, and community-generated performance data. This empowers gamers to make informed decisions about which titles to explore based on their hardware capabilities.

**FPS Submission:**

Gamers can submit their frames per second (FPS) benchmarks for specific games and graphics presets, allowing them to share and compare their performance metrics with fellow users. This feature fosters a sense of community and healthy competition among players.

# Different auth0 flows

## Authorization Code Flow (with PKCE):

The Authorization Code Flow with Proof Key for Code Exchange (PKCE) begins with the client application initiating the authentication process by redirecting the user to the Auth0 Authorization Server's /authorize endpoint. This redirect includes parameters such as the client ID, redirect URI, requested scopes, and a code verifier generated by the client. Upon successful authentication, Auth0 redirects the user back to the client application's redirect URI with an authorization code. The client then exchanges this authorization code and the original code verifier for an access token and, optionally, a refresh token by making a POST request to the /token endpoint. Auth0 verifies the authorization code and code verifier before issuing the tokens, providing a secure authentication mechanism suitable for web applications. This token then gets send to the backend service which will verify it.

## Implicit Flow (Legacy):

The Implicit Flow is a legacy authentication mechanism typically used for JavaScript-based applications unable to securely store client secrets. In this flow, the client application redirects the user to the Auth0 Authorization Server's /authorize endpoint, similar to the Authorization Code Flow. However, instead of receiving an authorization code, Auth0 directly returns an access token to the client application's redirect URI, typically embedded within the URL fragment. While straightforward, this flow is considered legacy due to security concerns related to exposing tokens in the browser, making it less recommended for modern applications.

## Resource Owner Password Credentials (ROPC) Flow:

The Resource Owner Password Credentials (ROPC) Flow allows a client application to directly exchange a user's credentials (username and password) for an access token. In this flow, the client application sends the user's credentials securely to the Auth0 Authorization Server's /token endpoint. Auth0 validates these credentials and issues an access token if authentication is successful. However, this flow should be used sparingly and only when other flows are not feasible, as it involves handling and transmitting sensitive user credentials, potentially increasing security risks.

## Conclusion

For my application I will be using the Authorization flow with PKCE, I will use this flow because it is the recommended flow by auth0, and will work well in my application. Auth0 also has integration with a lot of frameworks using this Authorization flow and Angular is one of them.

# Integration in angular

Auth0 is integrated in to angular with the Auth0 library provided by auth0. You can find this library on the official [auth0 github](https://github.com/auth0/auth0-angular). This library makes it very easy to integrate the angular application with auth0. You need to specify a few fields like:

* Domain: Domain is the Domain of you auth0 client
* ClientId: The clientId of you auth0 application
* AuthorizationParams: Used to tell auth0 what the application should be able to access, for example here we include an audience, this audience in our case is the audience of our api application, this means that our auth0 client can access the api.
* Redirect\_uri: This is the URL the application will redirect back to
* http Interceptor: This interceptor will add our access token to our request automatically be intercepting the request and adding the authorization header.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving

## Integration with KrakenD

Integration auth0 with KrakenD is very simple. KrakenD has auth0 support build-in. I just need to tell the application where to get the signature for verification and the audience it needs to verify.

