UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIA DA COMPUTAÇÃO

Brendon Vicente Rocha Silva

Implementação do protocolo *OpenID Connect for Identity Assurance 1.0* no *Keycloak*

Florianópolis
2023

Brendon Vicente Rocha Silva

Implementação do protocolo *OpenID Connect for Identity Assurance 1.0* no *Keycloak*

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Ciência da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Ricardo Felipe Custódio, Dr.
Coorientador: Prof. Frederico Schardong, Me.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Silva, Brendon Vicente Rocha
Implementação do protocolo OpenID Connect for Identity Assurance 1.0 no Keycloak / Brendon Vicente Rocha Silva ; orientador, Ricardo Felipe Custódio, coorientador, Frederico Schardong, 2023.
50 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciências da Computação. 2. Identidade eletrônica. 3. Segurança computacional. 4. OpenID Connect. 5. Keycloak. I. Custódio, Ricardo Felipe. II. Schardong, Frederico. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Brendon Vicente Rocha Silva

Implementação do protocolo *OpenID Connect for Identity Assurance 1.0* no *Keycloak*

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo curso de Graduação em Ciência da Computação.

Florianópolis, 13 de novembro de 2023.

Prof. Lúcia Helena Martins Pacheco, Dra.
Coordenadora do Curso

Banca Examinadora:

Prof. Ricardo Felipe Custódio, Dr.
Orientador
Universidade Federal de Santa Catarina

Prof. Frederico Schardong, Me.
Coorientador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
Universidade Federal de Santa Catarina

Maurício de Vasconcelos Barros
Avaliador
Universidade Federal de Santa Catarina

Este trabalho é dedicado à minha namorada linda e a Deus, que não me deu 5 cm a mais de altura, mas me deu um grande... coração.

AGRADECIMENTOS

Agradeço às pessoas que possibilitaram o desenvolvimento desse trabalho e me fizeram ser quem eu sou, sem vocês nada disso teria sido feito. Agradeço aos meus pais e minha família, por me apoiarem sempre e por terem pavimentado o caminho de decisões e escolhas para que eu pudesse chegar até onde cheguei. Agradeço à minha namorada, por ter me auxiliado no desenvolvimento desse trabalho mais do que seria moralmente aceitável e por ter segurado a barra nesse semestre infernal. Agradeço ao meu orientador e coorientador, pela parceria, por terem me introduzido nesse mundo acadêmico e por terem me dado tantas oportunidades de crescimento antes e durante o desenvolvimento desse trabalho. Por fim, agradeço ao Laboratório de Segurança em Computação da UFSC (LabSEC) e ao Operador Nacional do Registro Civil de Pessoas Naturais (ONRCPN) por terem aberto tantas portas nessa jornada e por terem proporcionado tamanho crescimento pessoal e profissional.

Também agradeço todo mundo que não foi citado acima, mas se encaixa entre pessoas que eu gosto (vocês sabem quem vocês são). Valeu, galera! Vocês são fera :)

As maquinações internas da minha mente são um enigma.
- Patrick Estrela

RESUMO

Tendo em vista o avanço acelerado do mundo digital, é necessário que as relações nesse meio sejam aprimoradas, de maneira a adaptá-las a um ambiente cada vez mais virtual. Um dos grandes desafios, nesse sentido, diz respeito ao processo de identificação digital: é cada vez mais complexa a tarefa de assegurar a identidade de indivíduos em um cenário virtual, bem como a de garantir a confiabilidade de suas informações nesse contexto. Foi pensando em solucionar desafios como este que surgiram, na última década, protocolos como o *OpenID Connect* 1.0. Esse protocolo, voltado ao campo da segurança computacional, busca permitir a identificação de usuários e a verificação de seus processos de autenticação, em um meio virtual. Seu objetivo é a padronização e viabilização de troca de informações, em um contexto seguro, acerca de indivíduos autenticados. Com a perspectiva do ritmo acelerado em que soluções tornam-se obsoletas no cenário computacional, o *OpenID Connect* 1.0 permite que extensões à especificação sejam desenvolvidas e propostas para que o protocolo mantenha-se atualizado. Uma dessas extensões mais recentes, intitulada *OpenID Connect for Identity Assurance* 1.0, pretende viabilizar a verificabilidade de informações e atributos de identidades eletrônicas, incorporando novos campos nas trocas de mensagens estabelecidas através do protocolo. A adoção dessa solução pode trazer benefícios a inúmeros sistemas, mas, por ser fruto de estudos recentes, pouco foi desenvolvido acerca de sua implementação; o protocolo ainda não foi implantado em um número expressivo de sistemas e sua utilização por serviços reais de gestão de identidade eletrônica ainda não foi consolidada. Nesse sentido, o presente trabalho propõe a materialização da especificação, em sua plenitude, através da implementação de um módulo funcional de código, que garante suporte ao protocolo no *Keycloak* — um dos provedores de identidade *no code* de código aberto mais proeminentes do mercado.

Palavras-chave: *OpenID Connect*. Identidade eletrônica. *Keycloak*.

ABSTRACT

In view of the accelerated advancement of the digital world, it is necessary to enhance relationships in this environment, adapting them to an increasingly virtual setting. One of the significant challenges in this regard pertains to the process of digital identification: ensuring the identity of individuals in a virtual scenario and guaranteeing the reliability of their information in this context has become an increasingly complex task. With the aim of addressing challenges like these, protocols such as OpenID Connect 1.0 have emerged in the last decade. This protocol, focused on the field of computer security, aims to enable user identification and verification of their authentication processes in a virtual environment. Its goal is the standardization and facilitation of secure information exchange regarding authenticated individuals. Given the perspective of the rapid pace at which solutions become obsolete in the computational landscape, OpenID Connect 1.0 allows for the development and proposal of extensions to the specification to keep the protocol up-to-date. One of these recent extensions, titled OpenID Connect for Identity Assurance 1.0, aims to enable the verifiability of information and attributes of electronic identities, incorporating new fields into the message exchanges established through the protocol. The adoption of this solution can bring benefits to numerous systems, but as it is the result of recent studies, little has been developed regarding its implementation; the protocol has not yet been deployed in a significant number of systems, and its use by real electronic identity management services has not been fully established. In this context, this work proposes the realization of the specification in its entirety through the implementation of a functional code module, ensuring support for the protocol in Keycloak—one of the most prominent open-source no-code identity providers in the market.

Keywords: OpenID Connect. Identity Assurance. Eletronic identity. Keycloak.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Desde sua idealização, em meados de 1960, até os dias atuais, a *internet* tornou-se um elemento fundamental no funcionamento da sociedade moderna (ABBATE, 2001). Hoje, é praticamente impossível imaginar-se em um mundo sem as influências dessa tecnologia; a *web* desempenha um papel crucial na maneira como nos comunicamos, trabalhamos, nos informamos ou até mesmo na forma em como nos divertimos.

Por conta do caráter veloz de sua evolução, a *internet* torna necessária a adaptação constante de tecnologias vigentes para que novos padrões de desempenho, usabilidade e segurança sejam atingidos. Novidades são descobertas em ritmo acelerado e tendências do mundo digital surgem a todo momento. Nesse sentido, sistemas e serviços incapazes de se adequarem às últimas inovações tecnológicas acabam ficando obsoletos.

Em um aspecto diretamente proporcional, o número de informações sensíveis guardadas no meio virtual têm aumentado exponencialmente, bem como a gravidade de potenciais fraudes e vazamentos. Nesse sentido, também é notável uma preocupação crescente com o uso, proteção e armazenamento de informações digitais. Um sinal explícito de tal fenômeno, por exemplo, é a promulgação de leis e regulamentos voltadas especificamente para segurança de dados digitais, como o Regulamento Geral sobre a Proteção de Dados (*General Data Protection Regulation* (GDPR)), na União Européia, e a Lei Geral de Proteção de Dados Pessoais (LGPD), no Brasil (European Commission, 2016; Brasil, 2018). Com esse contexto em mente, diversas estratégias foram desenvolvidas para garantir a confiabilidade em transações envolvendo identidades digitais e transferência de dados confidenciais *online*. Dentre os esforços nesse campo, contribuições de extrema relevância precisam ser citadas. Dentre elas, a criação de protocolos de segurança voltados à concessão de autorização digital e à manipulação de informações no contexto de identidades eletrônicas. Nesse cenário específico, podem ser citados alguns protocolos proeminentes, como o ***OAuth* 2.0** (HARDT, 2012) e o ***OpenID Connect* 1.0** (SAKIMURA et al., 2014a), que é vastamente utilizado em grandes sistemas e aplicações de renome ao redor do mundo (MAINKA et al., 2017; FETT; KÜSTERS; SCHMITZ, 2016).

Em suma, essas especificações buscam prover para os usuários que as utilizam mais segurança acerca de suas informações e como elas são gerenciadas e, para os sistemas que os implementam, a garantia de que um indivíduo é quem ele alega ser. Entretanto, mesmo com modelos direcionados para o contexto discutido, é necessário que estratégias de segurança sejam revisadas, reimaginadas e expandidas regularmente. Protocolos de segurança estão sujeitos a limitações por inúmeras causas, seja por um descuido no momento de sua idealização ou até mesmo por motivos que não poderiam ser previstos durante sua criação, como o uso exponencial de *smartphones* na última década ou a popularidade de aplicações *browser-based*, por exemplo (DENNISS; BRADLEY, 2017; PARECKI; WAITE, 2023).

É devido a esses aspectos que o *OpenID Connect* permite a integração de novas funcionalidades ao projeto, através da criação de extensões; uma dessas extensões é o ***OpenID Connect for Identity Assurance* 1.0** (LODDERSTEDT et al., 2022). Este módulo de extensão,

cuja última versão data agosto de 2022, trata da adição de novos campos, atributos e mecanismos que permitem atestar a veracidade de informações do usuário. Isso é feito com a finalidade de garantir a verificabilidade dos dados armazenados em sistemas que tratam da manipulação de informações sensíveis.

É importante salientar, no entanto, que por conta do caráter recente de sua publicação, a especificação concentra-se largamente no campo teórico, com poucas instâncias de aplicações práticas que a utilizem e, principalmente, poucas ferramentas que permitam sua plena implantação. Pensando nisso, é proposta uma materialização do protocolo; o objetivo é viabilizar o suporte ao *OpenID Connect for Identity Assurance 1.0* no ***Keycloak***, um projeto de código aberto que implementa o *OpenID Connect* e que atualmente configura-se como um dos provedores de identidade mais utilizados no campo (Keycloak, 2022). Esse suporte deve ser feito através do desenvolvimento de uma solução modular, que permita a implantação do protocolo em sistemas reais, já estabelecidos e funcionais. Dessa forma, espera-se ainda que a solução encontrada pavimente o caminho para que trabalhos futuros envolvendo o protocolo sejam realizados.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Desenvolver um módulo de extensão *open source* ao provedor de identidades *Keycloak*, capaz de garantir suporte a requisições, como descritas na especificação do protocolo *OpenID Connect for Identity Assurance 1.0*.

1.1.2 Objetivos Específicos

- Realizar um levantamento de implementações concretas existentes que aleguem suporte ao protocolo estudado;
- Analisar as soluções encontradas, de forma a determinar seus pontos fortes, estratégias de implementação e possibilidades de melhoria;
- Desenvolver uma solução própria, tendo em vista o conhecimento adquirido, que seja modular e que garanta, ao *Keycloak*, o suporte a repostas e requisições, como descritas na especificação do protocolo *OpenID Connect for Identity Assurance 1.0*;
- Conduzir a validação da solução e sua efetiva conformidade com o protocolo;
- Interpretar e discutir os resultados obtidos.

1.2 MOTIVAÇÃO E JUSTIFICATIVA

Por conta de ser um protocolo recente, ainda em desenvolvimento e projetado para casos de uso específicos, o *OpenID Connect for Identity Assurance 1.0* ainda não é suportado, em sua plenitude, por uma quantidade expressiva de identidades eletrônicas. Esse fenômeno é agravado, ainda, pela escassez de ferramentas que permitam o suporte ao protocolo em sistemas gerenciadores de identidades, como o *Keycloak*. Plataformas que desejem implementar as diretrizes descritas pela especificação, portanto, precisam desenvolver suas próprias soluções, o que é um processo custoso, burocrático e suscetível a erros.

O presente trabalho se faz útil, nesse contexto, buscando aprofundar discussões técnicas e teóricas acerca da implementação do protocolo, enquanto garante seu suporte em um dos *softwares* gerenciadores de identidades mais proeminentes do mercado, o *Keycloak*. Pretende-se criar precedentes para que o protocolo possa ser explorado, no futuro, e que possa ser utilizado por aplicações reais que já façam uso do *Keycloak*.

1.3 MÉTODO DE PESQUISA

Em um primeiro momento, uma revisão narrativa da literatura científica foi realizada, com o objetivo de revisitar conceitos ligados à segurança computacional, identidade eletrônica e ao protocolo estudado. O conhecimento adquirido nessa etapa serviu como base para o início do projeto e fundamentou as decisões tomadas nas fases seguintes.

Em seguida, um levantamento, também narrativo, de soluções existentes no mercado foi feito. Nesse estágio, buscaram-se implementações do protocolo em sistemas gerenciadores de identidade diversos, incluindo o *Keycloak*. As conclusões tomadas acerca dessa investigação tiveram relevância na definição do escopo do projeto e nortearam o desenvolvimento da solução proposta. Esse momento serviu para a identificação de lacunas não exploradas e pontos de melhoria em outros serviços, com a finalidade de contemplá-los no produto do presente estudo.

O efetivo suporte ao protocolo foi adicionado ao *Keycloak* por meio da implementação de *Service Providers* personalizados: módulos escritos em JAVA que viabilizam o desenvolvimento de funcionalidades extras ao programa, sem que seja necessário alterar seu código-fonte (Keycloak, 2023). Cumprindo, dessa maneira, com o requisito imposto de que a solução encontrada deveria ser modular, para que o código disponibilizado pudesse ser aplicado a casos de uso e sistemas reais já existentes.

Por fim, uma série de testes foi conduzida, com a finalidade de validação do projeto, de maneira a garantir sua conformidade com os comportamentos descritos pela documentação do protocolo. Essa etapa permitiu não só a discussão dos resultados e suas características, como consolidou a definição de trabalhos futuros.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho dispõe da seguinte estrutura:

- No Capítulo 2 serão discutidos conceitos cruciais para a compreensão dos tópicos abordados durante o desenvolvimento do trabalho. Esses conceitos também serão úteis para que se entenda a problemática e a motivação da utilização de identidades eletrônicas no contexto computacional, assim como a importância de autenticidade de dados transmitidos no meio virtual;
- O Capítulo 3 apresentará os resultados atingidos pelo levantamento de soluções do mercado, bem como as conclusões obtidas durante esse processo.
- No Capítulo 4, o processo de desenvolvimento utilizado para modelar a solução que garantirá suporte ao *OpenID Connect for Identity Assurance* 1.0 será apresentado. Nele serão detalhadas as ferramentas utilizadas, as estratégias de implementação adotadas e suas características e peculiaridades;
- O Capítulo 5 discorrerá acerca do produto da implementação e seus resultados;
- Por fim, o Capítulo 6 será responsável por expor as conclusões tomadas durante o desenvolvimento do projeto, retomando a motivação da elaboração deste trabalho e destacando pontos de interesse na solução proposta. Esse capítulo também apresentará os possíveis desenvolvimentos futuros referentes ao tema abordado.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo desse capítulo é prover conhecimentos necessários para o entendimento do conteúdo abordado neste trabalho. Aqui, serão discutidos os conceitos relacionados à identidade eletrônica, bem como os protocolos de segurança utilizados nesse contexto, além de um panorama acerca da ferramenta que foi foco de estudos durante o desenvolvimento do projeto.

2.1 IDENTIDADE ELETRÔNICA

O anseio por determinar o que caracteriza um indivíduo e o que é identidade permeia a humanidade desde os primórdios da civilização moderna. A definição de identidade é objeto de estudo na filosofia há muito tempo: conceitos como o paradoxo do navio de Teseu[1], que buscam questionar e incitar reflexões acerca da definição de individualidade, unicidade e permanência foram introduzidos séculos atrás e são temas de pesquisa até hoje (BROSS, 2019).

Ao se distanciar do campo filosófico, no entanto, a **identidade**, como conceito, pode ser definida como uma coleção única de características, atributos e informações — como nome próprio, endereço de *email*, senha, números de documentos, entre outros — que pode ser utilizada para representar um indivíduo ou uma organização (ISO Central Secretary, 2019). De forma intuitiva, ao se tratar de um ambiente virtual, a **identidade eletrônica** pode ser descrita como um conjunto de propriedades atreladas a uma entidade e armazenadas em um meio digital.

Não são raros os exemplos de identidades eletrônicas no cotidiano de usuários digitais. É improvável que alguém que utilize serviços eletrônicos regularmente nunca tenha se deparado com uma situação em que fosse necessário sua identificação e, portanto, seu cadastro. Nesse contexto, o próprio indivíduo deve fornecer seus dados e informações pessoais, de forma a gerar, no sistema que deseja identificá-lo, sua própria identidade. Nas seguintes interações com esse sistema, o usuário deverá declarar sua relação com a identidade criada, para que possa ser identificado.

Nesse sentido, esse processo de identificação (ou autenticação) consiste em atestar a associação entre uma entidade e sua identidade, normalmente com o auxílio de um **atributo identificador** (FERDOUS; NORMAN; POET, 2014). Atributos identificadores são informações únicas, que não se repetem em outras identidades, e têm a função de representar uma coleção de atributos. Um atributo identificador pode ser um número único em um documento, ou um nome de usuário em uma conta virtual, por exemplo.

A autenticação pode ocorrer de duas maneiras: formas de autenticação podem ser acordadas previamente; ou o envolvimento de uma terceira parte confiável pode ser requerido (SCHARDONG; CUSTÓDIO, 2022). Uma forma de autenticação acordada acontece, por exemplo, quando um indivíduo tem sua biometria atestada visualmente ao apresentar um cra-

[1] Em resumo, o paradoxo retratado pelo filósofo grego Plutarco descreve um navio cujas peças originais foram totalmente substituídas, pouco a pouco, ao longo do tempo. Ao fim, é questionado se o navio, após ter todas as suas peças trocadas, pode ser considerado o mesmo do início (LESTERJR.; LAMBDIN, 1998).

chá de identificação para entrar em um edifício ou quando alguém informa seu endereço de *email* e senha ao se autenticar em um *website*. Por outro lado, um exemplo de autenticação com envolvimentos de terceiros pode ser observado quando um indivíduo apresenta um documento de identidade a um estabelecimento ou quando um usuário se autentica perante um sistema utilizando uma de suas redes sociais. Nesse caso, há uma relação de confiança entre as partes envolvidas: o estabelecimento confia nas informações contidas na identidade, pois confia no órgão emissor do documento, bem como o sistema deve confiar nos mecanismos de identificação da rede social; e o indivíduo sendo identificado, de forma semelhante, confiou que suas informações pessoais estariam seguras ao cedê-las durante a criação da identidade.

No âmbito virtual, a entidade encarregada por armazenar e distribuir os dados do usuário, ao mesmo tempo em que se responsabiliza por autenticá-los e garantir a confidencialidade de suas informações, é o **provedor de identidades** (ou *Identity Provider* (IdP)). Ele é responsável por assegurar e prover dados da identidade de um usuário, assim como por garantir sua autenticidade (ISO Central Secretary, 2019).

A autenticidade desses dados pode ser atestada por meio de alguns mecanismos, como o **nível de garantia**. O nível de garantia de um usuário é calculado em função de parâmetros determinados pelo provedor de identidades e normalmente reflete com que precisão pode se afirmar que a identidade de um indivíduo pertence ao dono verdadeiro das informações (GRASSI; GARCIA; FENTON, 2017). Não é raro, por exemplo, em sistemas que manipulam informações sensíveis, deparar-se com uma organização em "níveis de usuário", como a classificação ouro, prata e bronze, utilizado pela plataforma digital de serviços públicos do governo brasileiro[2], que faz uso dos métodos de autenticação biométrica cadastrados para cada cidadão (Ministério da Gestão e da Inovação em Serviços Públicos, 2021b). Esses níveis permitem que aplicações que se conectem ao provedor de identidades possam gerenciar melhor as permissões de cada indivíduo. Desta forma, um usuário nível bronze, por exemplo, pode ser impedido de realizar operações sensíveis, já que não é possível atestar com alto nível de certeza que a pessoa utilizando o sistema é quem alega ser, enquanto um usuário nível ouro não tem restrições.

Por fim, todas as identidades, sejam elas físicas ou eletrônicas, devem ser passíveis de verificação de propriedade; ou seja, devem permitir que sejam reconhecidas e providenciar mecanismos para que seja possível distinguir e autenticar os usuários aos quais elas pertencem (KIENNERT; BOUZEFRANE; THONIEL, 2015).

2.2 OAUTH 2.0

Considerando o crescimento considerável das relações digitais e o aumento da complexidade de transações realizadas no meio virtual, providências foram tomadas, ao longo do tempo, para aumentar a confiabilidade e segurança das operações efetuadas na *Internet*. A família de protocolos *OAuth*, nesse cenário, surgiu como uma ferramenta para auxiliar na troca de

[2] O sistema *gov.br* permite que usuários tenham acesso digital a serviços públicos digitais brasileiros (Ministério da Gestão e da Inovação em Serviços Públicos, 2021a).

informações sigilosas através do Protocolo de Transferência de Hipertexto (ou *Hypertext Transfer Protocol* (HTTP))[3] (HARDT, 2012; HAMMER-LAHAV, 2010). O ***OAuth 2.0*** é, portanto, a segunda versão da família de protocolos, criado com o objetivo de permitir que aplicações operando na *Internet* obtenham acesso a recursos privados de usuários ou entidades, mediante autorização (HARDT, 2012).

Antes que se elabore o funcionamento, ou até a motivação do uso dessa ferramenta, é importante que sejam esclarecidos alguns papéis e jargões utilizados pelo protocolo: o **cliente** é uma aplicação que deseja consumir informações sigilosas de determinado usuário; nesse cenário, o **proprietário do recurso** é o dono dos dados requisitados; tais recursos devem estar armazenadas em um **servidor de recursos**; por fim, o **servidor de autorização** (ou provedor de identidades) se encarrega de autenticar o usuário e autorizar que o cliente acesse as informações desejadas (SCHARDONG et al., 2022). Embora os servidores de recursos e autorização sejam tratados, na documentação, como entidades independentes, em aplicações concretas e situações específicas é possível que ambos representem a mesma entidade (HARDT, 2012).

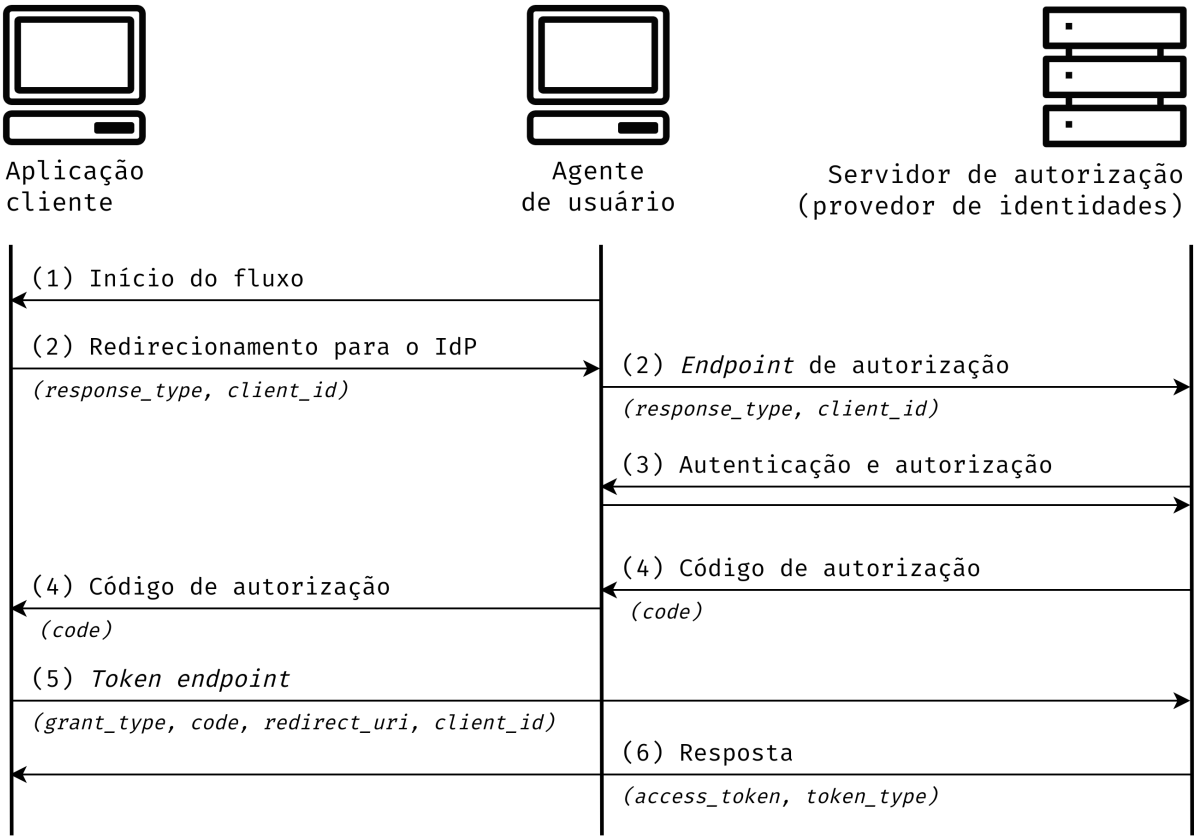
A família de protocolos em questão se faz útil pois, em sistemas que implementam o modelo de autenticação cliente-servidor tradicional — ou seja, não empregam as práticas descritas pelo *OAuth* — para que o usuário conceda a uma aplicação permissão para consumir informações armazenadas em um servidor externo, é necessário informá-la de suas credenciais (ARGYRIOU; DRAGONI; SPOGNARDI, 2017). Ao fazer isso, o proprietário do recurso perde o controle de acesso aos seus dados, já que o cliente se autentica no servidor com os mesmos privilégios do dono da identidade. Uma aplicação má intencionada, nessa situação, pode não só obter informações sem permissão, como modificá-las ou excluí-las (HARDT, 2012).

Para que um cliente autentique-se com o servidor de recursos, nos princípios descritos pelo *OAuth 2.0*, é necessário que uma credencial especial seja apresentada: um ***Token* de Acesso** (ou *Access Token*) (FOTIOU et al., 2020). Esse *token* é gerado pelo servidor de autorização, mediante autenticação e permissão do usuário; ele é um conjunto de caracteres que permite que dados privados sejam consumidos, modificados ou excluídos, por um período de tempo pré-determinado (HARDT, 2012). Além das credenciais de acesso, um *token* pode conter, imbuído em sua estrutura, informações importantes, como validade e escopo das permissões de acesso. Um *token* pode, dessa forma, conceder permissão para que um cliente apenas leia dados específicos, por exemplo.

Para que se obtenha um *token*, o protocolo descreve alguns fluxos de comunicação possíveis. No entanto, apenas um deles é recomendado para uso em aplicações cuja segurança é um fator a ser considerado, já que outros fluxos são descritos pela documentação com a intenção de facilitar a transição para o *OAuth 2.0*, em sistemas que adotam outros modelos de autenticação, ou tiveram o uso desencorajado depois que estudos apontaram possíveis vulnerabilidades (SCHARDONG et al., 2022). Na Figura 1, são descritos os passos e trocas de mensagens realizados durante o único fluxo de comunicação recomendado, chamado de concessão de código

[3] O *Hypertext Transfer Protocol* é um protocolo de comunicação utilizado amplamente para a transferência de dados na *web* (NIELSEN et al., 1999)

Figura 1 – *Authorization Code Grant* como descrito no protocolo *OAuth 2.0*.



Fonte: O autor.

de autorização, ou ***Authorization Code Grant***.

Primeiramente, o usuário dá início ao fluxo, sinalizando que deseja compartilhar um recurso privado com uma aplicação. Uma situação prática desse cenário pode ser vista ao tentar acessar uma página restrita, ou ao escolher autenticar-se com as redes sociais em um *site*, por exemplo (SCHARDONG et al., 2022).

No passo seguinte, o cliente redireciona o usuário para o endereço de autenticação do provedor de identidades, em conjunto com dois parâmetros obrigatórios (outros parâmetros opcionais podem ser fornecidos, mas serão abordados mais adiante): `response_type`, que contém o valor `token`, indicando que o fluxo atual trata-se do tipo *Authorization Code Grant*; e `client_id`, contendo a identificação do cliente, perante o servidor de autorização (HARDT, 2012). Esse redirecionamento fica sob responsabilidade do agente de usuário[4] do proprietário do recurso a ser compartilhado.

Em seguida, o usuário é autenticado pelo provedor de identidades e, em alguns casos, é necessária a autorização explícita do proprietário para que suas informações sejam compartilhadas (em outros casos, a autorização implícita é suficiente). A documentação do protocolo não define como deve ser feita a autenticação de usuários; usualmente são solicitadas credenciais,

[4] Um agente de usuário é qualquer *software* encarregado de recuperar e apresentar conteúdos *web*, como um navegador ou aplicativo (FIELDING; NOTTINGHAM; RESCHKE, 2022).

como nome de usuário e senha (SCHARDONG et al., 2022).

No quarto passo, o proprietário do recurso é redirecionado, novamente por meio de seu agente de usuário, para a aplicação cliente. Se a autenticação do passo anterior foi bem sucedida, um **código de autorização**, com curta validade, é gerado pelo servidor de autorização; esse código é repassado na requisição através de um parâmetro `code`. Caso uma falha aconteça, no entanto, o provedor notifica o usuário através do parâmetro `error`.

Em posse do código de autorização, o cliente é capaz de enviar uma requisição diretamente ao provedor de identidades; nela, os seguintes parâmetros deverão estar presentes: `grant_type`, cujo valor será `authorization_code`; `code`, contendo o código de autorização previamente adquirido; e `client_id`, com o mesmo valor do passo dois, caso o cliente não precise ser autenticado. No *OAuth 2.0*, entretanto, clientes *web* devem realizar alguma forma de autenticação mediante o servidor de autorização (SCHARDONG et al., 2022). Embora a documentação não especifique maneiras de realizar tal autenticação, há recomendações de protocolos específicos para realizar esse processo, que não cabem ao escopo do presente trabalho (LODDERSTEDT et al., 2023; SAKIMURA et al., 2014c).

Por fim, o provedor de identidades verifica se o código de autorização recebido é válido e foi emitido para a aplicação responsável pela requisição (SCHARDONG et al., 2022). Nesse caso, um novo *token* de acesso é gerado e enviado para o cliente, através do parâmetro `access_token`, em conjunto com o tipo do *token* utilizado (`token_type`) (HARDT, 2012). Munida desse *token*, a aplicação pode, então, finalmente consumir os dados desejados.

2.3 OPENID CONNECT 1.0

Mesmo sendo abrangente em sua especificação, o *OAuth 2.0* deixa de lado alguns aspectos importantes; o protocolo, por exemplo, não provê meios para que o cliente adquira informações acerca da autenticação dos usuários (SCHARDONG et al., 2022). Pensando em solucionar esse e outros problemas, foi proposta a criação de um novo protocolo voltado especificamente para soluções relacionadas à identidade eletrônica: o ***OpenID Connect 1.0*** (SAKIMURA et al., 2014a), que implementa uma camada de identidade sobre a segunda versão do *OAuth* e permite que aplicações autentifiquem usuários e/ou obtenham informações básicas acerca de seus perfis (DOMENECH; COMUNELLO; WANGHAM, 2014).

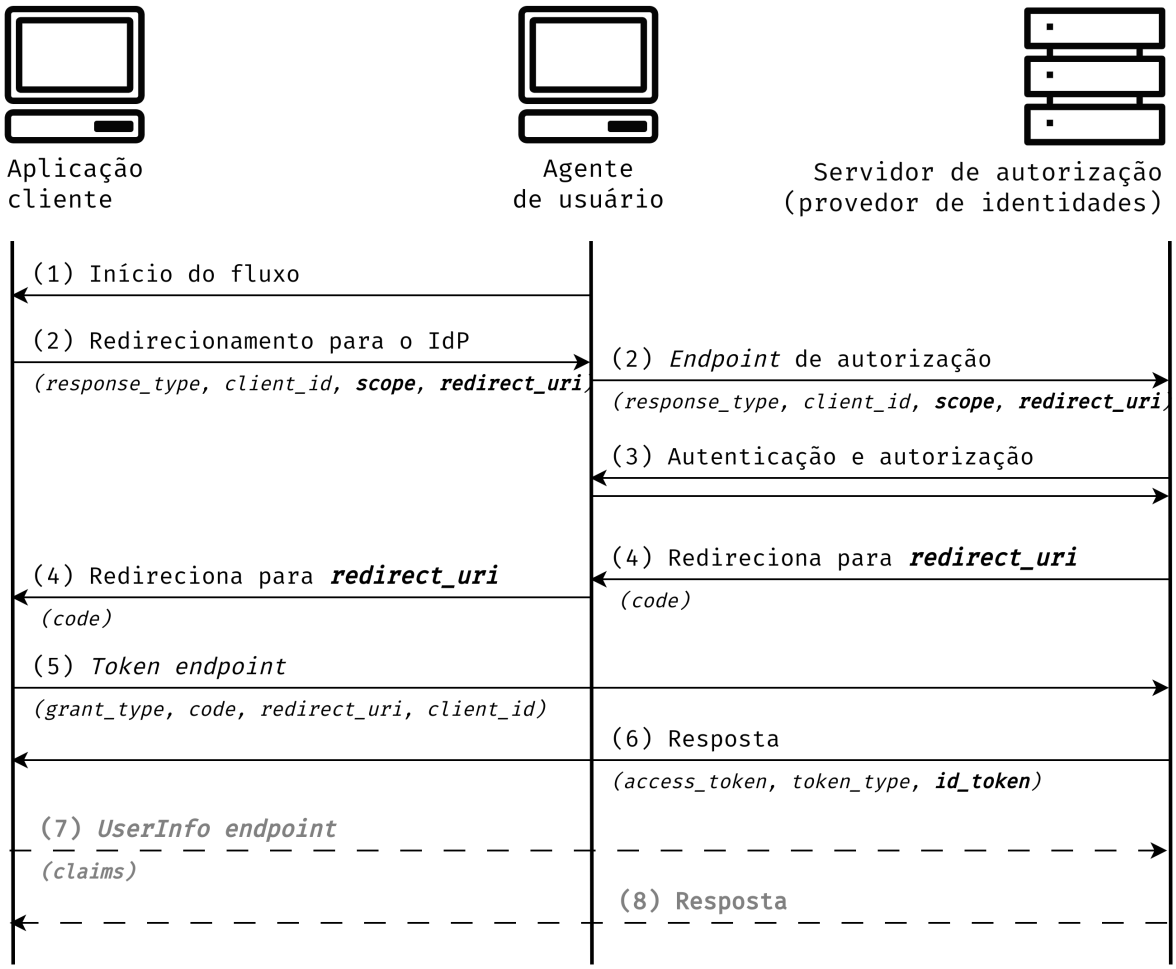
As informações de autenticação e atributos de usuário são recebidas pela aplicação cliente através de objetos na notação *JavaScript Object Notation* (JSON) [5], podendo ou não estar assinados e/ou encriptados, em um JSON *Web Token* (JWT)[6] (NAVAS; BELTRÁN, 2019; SAKIMURA et al., 2014a). Essas informações e atributos recebem o nome de Reivindicações (ou ***Claims***, no texto original) (SAKIMURA et al., 2014a).

[5] JSON refere-se a uma forma de serializar dados estruturados, em texto, utilizando o formato de objetos da linguagem *JavaScript* (BRAY, 2017).

[6] O JSON *Web Token* é uma forma de representar dados assinados e/ou criptografados, cujo *payload* contém um objeto JSON. Os *tokens* são codificados no padrão *Base64* e podem ser assinados utilizando um segredo privado ou um esquema de chaves pública/privada (JONES; BRADLEY; SAKIMURA, 2015).

O *OpenID Connect 1.0* modifica o *OAuth 2.0* de forma aditiva. Isso significa que novas funcionalidades são implementadas ao protocolo base sem que as originais sejam removidas. Para isso, o *OpenID Connect* faz uso de parâmetros opcionais do *OAuth 2.0* enquanto incorpora novos, ao mesmo tempo instruindo que parâmetros não reconhecidos pela documentação sejam ignorados (SCHARDONG et al., 2022). Na Figura 2, são abordadas adições ao fluxo de concessão de código de autorização, como descrito pelo protocolo, em relação ao *OAuth 2.0*.

Figura 2 – *Authorization Code Grant* no *OpenID Connect 1.0*. Em negrito, passos e parâmetros adicionais em relação ao fluxo descrito pelo *OAuth 2.0*. Passos opcionais são apresentados em cor acinzentada.



Fonte: O autor.

Em primeiro lugar, o parâmetro `scope`, opcional no fluxo anterior, passa a ser obrigatório. Ele define o escopo do acesso requisitado e, tratando-se do protocolo *OpenID Connect 1.0*, deve conter, entre outros, o valor `openid` (HARDT, 2012; SAKIMURA et al., 2014a). Quatro novos escopos opcionais são especificados pela documentação: `profile`, `email`, `phone` e `address` (SAKIMURA et al., 2014a). Cada um se encarrega de solicitar dados do usuário, de acordo com o conjunto de atributos disponíveis (conforme Apêndice A).

As *claims* requisitadas no escopo da requisição são entregues durante o processo de obtenção do *token* de acesso, através de um novo argumento obrigatório, chamado `token_id` (SCHARDONG et al., 2022). Ele é responsável pela transmissão de um *token* de identificação

(ou ***ID token***) que contém atributos relacionados ao processo de autenticação do usuário, podendo conter eventuais informações relacionadas à sua identidade (SAKIMURA et al., 2014a). Adicionalmente, algumas *claims* obrigatórias em *tokens* de identificação são descritas pela documentação: (i) `iss`, identificando o emissor do *token*; (ii) `sub`, contendo um identificador do usuário, perante o servidor de autorização; (iii) `aud`, com informações sobre a quem esse *token* é destinado; (iv) `exp`, com a validade do *token*; (v) `iat`, com sua data e hora de emissão; (vi) `auth_time`, contendo a data e hora da autenticação do usuário; (vii) e `nonce`, uma *string* utilizada para associar um cliente a um *token* de identificação. Outras *claims* opcionais são mencionadas, como: (i) `acr`, que é utilizada para indicar o nível de garantia do processo de autenticação, de acordo com os padrões estabelecidos pela ISO/IEC 29115 (ISO, 2013); (ii) e `amr`, que contém um *array* de objetos JSON, indicando os métodos de autenticação utilizados.

Outra mudança refere-se ao uso do parâmetro `redirect_uri`: este parâmetro, agora obrigatório, era considerado opcional em implementações do *OAuth* (HARDT, 2012). Ele especifica o endereço da aplicação cliente, para o qual usuários devem ser redirecionados, após autenticação; seu valor deve ser o mesmo informado durante o registro da aplicação, mediante o provedor de identidades (SAKIMURA et al., 2014a).

Por fim, o cliente, utilizando o protocolo *OpenID Connect 1.0*, é capaz de solicitar informações do usuário *a posteriori*, conforme necessidade. Isso pode ser feito através de requisições direcionadas à um *endpoint* especial do servidor de autorização, chamado *UserInfo*. Para isso, o cliente deve informar os atributos que deseja consumir, através do parâmetro `claims`, que transmite um objeto JSON contendo *claims* indicando quais dados (conforme Apêndice A) são relevantes. O servidor deve, então, retornar as informações requisitadas, no corpo da resposta (SAKIMURA et al., 2014a).

O *OpenID Connect 1.0*, no entanto, não provê meios para que clientes possam obter informações referentes às configurações de provedores de identidade, como endereços de *endpoints* ou listas de funcionalidades e atributos suportados (SCHARDONG et al., 2022). Uma especificação suplementar foi desenvolvida, então, com o intuito de abordar esses e demais pontos: o *OpenID Connect Discovery 1.0* (SAKIMURA et al., 2014b). Essa extensão ao *OpenID* define, entre outros, que o endereço `/.well-known/openid-configuration` seja utilizado para disponibilizar um objeto JSON com configurações pertinentes do provedor.

2.4 OPENID CONNECT FOR IDENTITY ASSURANCE 1.0

O *OpenID Connect for Identity Assurance 1.0* é uma extensão ao *OpenID Connect 1.0*, proposta em março de 2019 (LODDERSTEDT; FETT, 2019). Hoje, o projeto, ainda em andamento, está sob desenvolvimento pelo *eKYC & IDA Working Group* — um grupo de pesquisa dentro da *OpenID Foundation*, que criou o *OpenID Connect*, com o objetivo de produzir extensões que padronizem a comunicação de informações acerca do nível de garantia de identidades eletrônicas (OpenID Foundation, 2022; LODDERSTEDT et al., 2022).

O propósito dessa extensão é permitir que dados a respeito do nível de garantia de

informações do usuário (o que, como, quando, de acordo com quais regras, usando quais evidências) sejam transmitidos, padronizados e suportados pelo protocolo (SHARIF et al., 2022). Essa funcionalidade é particularmente útil em aplicações que requerem alto grau de segurança, tendo em vista, por exemplo, regulamentações emergentes abrangendo segurança de dados, como a LGPD e GPDR (Brasil, 2018; European Commission, 2016).

Um provedor de identidades, ao implementar o *OpenID Connect for Identity Assurance 1.0*, deve ser capaz de fornecer informações relativas ao processo de validação dos dados de um usuário, como regulamentação sob a qual atua e sob a qual os dados foram validados, bem como evidências quanto à integridade dessas informações (LODDERSTEDT et al., 2022). É possível, por exemplo, informar a uma aplicação cliente que determinado usuário possui nome verificado com certo nível de garantia (de acordo com alguma especificação utilizada), evidenciado por seu documento de identidade enviado em anexo.

Algumas leis, especificações e regulamentações são citadas pela documentação e referenciadas como estruturas de confiança (ou ***trust frameworks***, no texto original) (LODDERSTEDT et al., 2022). No geral, esses *frameworks* definem uma série de parâmetros e requisitos que devem ser considerados durante transações relacionadas à identidade eletrônica, como, por exemplo, níveis de garantia e seus respectivos critérios, ou processos de validação de atributos do usuário. Dentre *frameworks* notáveis, utilizados nesse contexto, é possível citar: a documentação NIST-800-63A, que estabelece os níveis de garantia IAL1, IAL2 e IAL3 (GRASSI et al., 2017); e o Sistema Europeu de Reconhecimento de Identidades Eletrônicas (ou eIDAS), responsável por regulamentar processos de identificação eletrônica na União Europeia (European Parliament and Council of the European Union, 2014).

Para se adequar ao máximo de jurisdições possíveis, o protocolo introduz algumas novas *claims* relacionadas às informações dos proprietários de identidades (descritas no Apêndice B) (LODDERSTEDT et al., 2022). A documentação propõe, ainda, um novo atributo, relacionado a processos de auditoria referentes à autenticação de usuários: o campo `txn` se encarrega de transmitir um *token* de eventos de segurança, que define identificadores e informações importantes, pertinentes a transações envolvendo identidades eletrônicas (HUNT et al., 2018; LODDERSTEDT et al., 2022).

Claims verificadas são entregues a aplicações cliente por meio de *tokens* de identificação, *tokens* de acesso e/ou requisições ao servidor, direcionadas ao *endpoint* de *UserInfo* (LODDERSTEDT et al., 2022). O Algoritmo 1 apresenta a estrutura básica da representação de informações verificadas, de acordo com o protocolo.

As informações validadas são propositalmente isoladas dos demais dados, com a finalidade de facilitar sua distinção, e são representadas através de um elemento JSON, chamado `verified_claims`. Esse elemento é composto por dois subcomponentes: `verification`, um objeto JSON que envolve informações acerca do processo de verificação dos dados; e `claims`, contendo os dados verificados propriamente ditos (conforme Apêndice A e Apêndice B).

O objeto `verification` contém, obrigatoriamente, o campo `trust_framework`, indicando o *framework* de confiança sob o qual os dados foram coletados, verificados e arma-

Algoritmo 1 – Representação de *claims* verificadas no *OpenID Connect for Identity Assurance 1.0*.

```
{
    "verified_claims": {
        "verification": {
            "trust_framework": "trust_framework_example"
        },
        "claims": {
            "given_name": "Max",
            "family_name": "Meier"
        }
    }
}
```

Fonte: Lodderstedt et al. (2022)

zenados; tendo isso em vista, campos opcionais são definidos pela documentação e podem ser utilizados para que o processo de validação das informações esteja em conformidade com os requisitos especificados pelo mesmo. Determinado *framework*, por exemplo, pode garantir que dados, sob um certo nível de garantia (especificado pelo campo opcional `assurance_level`), não precisam de provas para serem considerados confiáveis, enquanto outro pode instruir que certas informações devem ser apresentadas junto a documentos oficiais de um usuário para serem legitimadas.

O protocolo permite que evidências variadas sejam apresentadas como forma de garantia e validação de dados transmitidos. Os tipos de comprovações, especificados na documentação, são: (i) `document`, compreendendo qualquer tipo de documento, físico ou eletrônico, pertencente ao usuário; (ii) `electronic_record`, representando dados digitais, obtidos por meios reconhecidos pelo *framework* de confiança; (iii) `vouch`, referenciando alguma entidade confiável, que atesta a confiabilidade dos dados; (iv) e `electronic_signature`, referindo-se a assinaturas eletrônicas.

Dados verificados podem ser requisitados por aplicações clientes de maneira similar ao processo feito para solicitar informações do usuário, em provedores implementando o protocolo *OpenID Connect 1.0* (SAKIMURA et al., 2014a). Esses dados podem ser exigidos de acordo com um escopo pré-determinado, ou através do parâmetro *claims*, adicionando à requisição um objeto JSON `verified_claims`, para determinar quais atributos do usuário são relevantes, em conjunto com quais processos de validação são desejáveis. Durante o processamento de respostas às requisições, o provedor de identidades deve ignorar mensagens que não entende e omitir *claims*, caso não as possua ou caso não atendam às exigências referentes ao processo de validação (LODDERSTEDT et al., 2022).

Enfim, o protocolo define uma série de novos atributos, relacionados aos processos

de validação suportados pelo servidor, que devem ser adicionados à lista de configurações do provedor (através do objeto JSON, disponível em `/.well-known/openid-configuration`), conforme descrito pelo *OpenID Connect Discovery 1.0* (LODDERSTEDT et al., 2022; SAKIMURA et al., 2014b).

2.5 KEYCLOAK

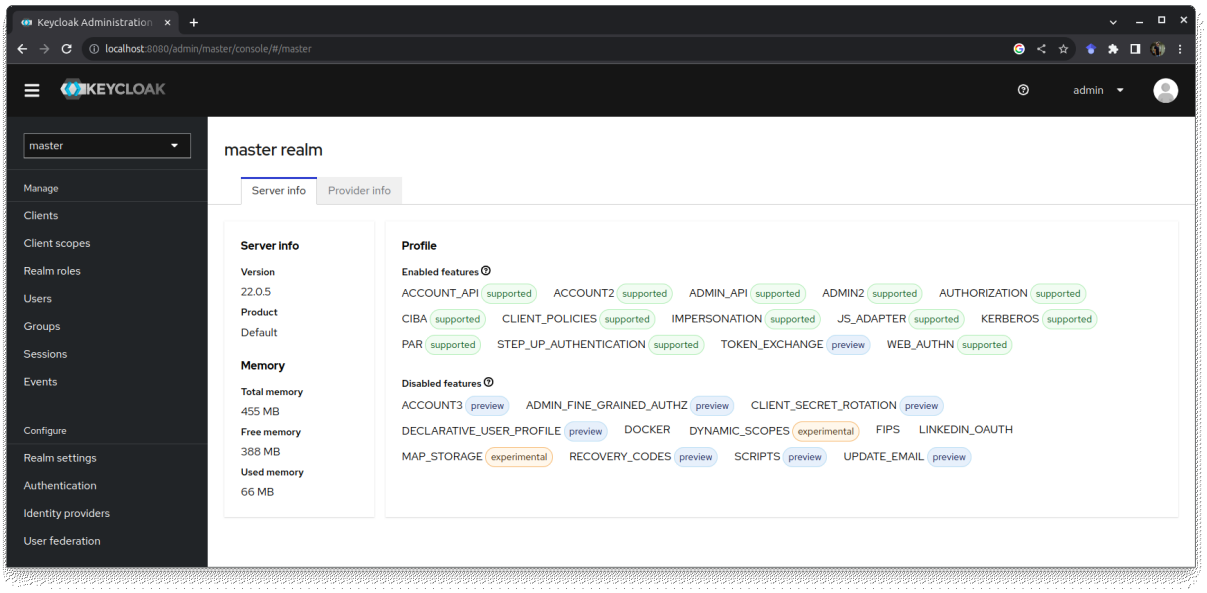
Ao se tratar do conceito de identidade eletrônica, um dos modos que pode ser destacado em seu uso atual é o ***single sign-on*** (ou SSO), onde o usuário autentica-se com uma única identidade em diversos sistemas (CLERCQ, 2002). Usualmente, essas identidades estão armazenadas em provedores terceiros, fora do domínio das aplicações que as consomem. No entanto, é incomum que sistemas utilizem apenas um método de cadastro/autenticação, ou que provedores de identidade genéricos possuam todas as informações necessárias ao caso de uso específico de cada aplicação.

Uma loja virtual, por exemplo, pode desejar que usuários cadastrem-se no sistema utilizando suas redes sociais, mas precisar de informações que não estejam inclusas em suas identidades, como endereço de moradia ou código de fidelidade da loja. Além disso, não se pode assumir que todos os clientes possuam uma rede social válida para realizar o cadastro. A loja poderia, nesse caso, decidir desenvolver seu próprio provedor de identidades, com os atributos necessários para o modelo de negócio, e que permitisse a integração com outros provedores.

Embora essa saída possa parecer lógica, o desenvolvimento de um sistema com esse nível de complexidade pode ser árduo e custoso. Dentre inúmeros fatores, que dificultam a implementação dessa solução, destacam-se a necessidade de: criar fluxos complexos de usabilidade (*login*, cadastro, recuperação de credenciais, entre outros); seguir diversos padrões e protocolos de segurança e comunicação, relacionados a identidades eletrônicas; desenvolver um sistema robusto, capaz de suportar um número expressivo de usuários; e criar um sistema sem falhas, já que possíveis vulnerabilidades e vazamentos de dados podem significar o comprometimento de informações críticas para o usuário (THORGERSEN; SILVA, 2021).

Para solucionar esses e outros problemas, *softwares* como o ***Keycloak*** foram desenvolvidos; o programa surgiu em 2014, com o propósito de facilitar o controle de identidades e acesso em aplicações de pequeno, médio e altíssimo porte (THORGERSEN; SILVA, 2021). O *Keycloak* é uma ferramenta de código aberto, escrita em JAVA, de alta performance, focada em aplicações modernas e capaz de ser implantado sem que linhas de código precisem ser escritas (DIVYABHARATHI; CHOLLI, 2020; Keycloak, 2013). A Figura 3 retrata a tela inicial do *Keycloak* e lista algumas de suas funcionalidades.

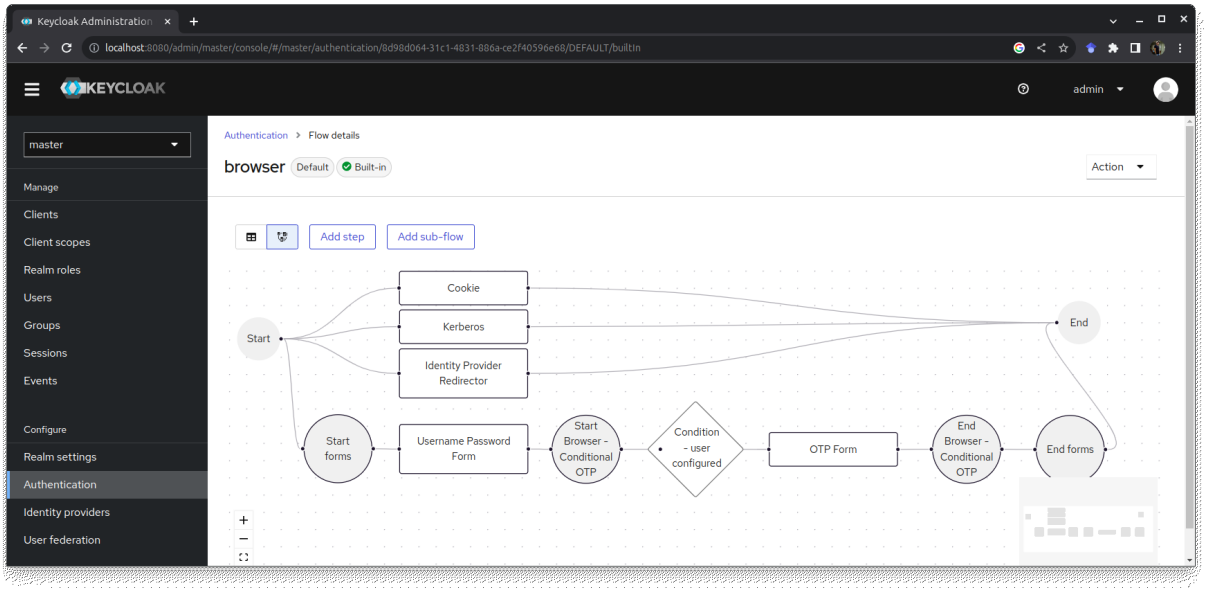
Figura 3 – Tela inicial do *Keycloak*.



Fonte: O autor.

As grandes vantagens da utilização desse *software*, se comparado à abordagem anterior, são a praticidade e segurança. O *Keycloak* deixa à disposição páginas totalmente customizáveis, relacionadas à autenticação, assim como diversos fluxos personalizáveis pré-definidos, como recuperação de credenciais, validação de atributos, *login* com múltiplos fatores de autenticação, entre outros (THORGERSEN; SILVA, 2021). Todos esses fluxos podem ser modificados sem que nenhum tipo de código precise ser escrito, a própria interface do sistema permite que seus comportamentos sejam alterados com apenas alguns cliques, como ilustrado pela Figura 4.

Figura 4 – Personalização do fluxo de autenticação de usuários através do *Keycloak*.



Fonte: O autor.

Além disso, a ferramenta se propõe a dar suporte a protocolos bem estabelecidos no contexto de segurança digital, como os supracitados *OAuth* 2.0 e *OpenID Connect* 1.0 (Keycloak, 2013). O *Keycloak* destaca-se, ainda, por ser um produto verificado e certificado pela *OpenID Foundation*, entidade encarregada da administração do protocolo *OpenID Connect* (OpenID Foundation, 2023).

É possível, ainda, que novas funcionalidades sejam projetadas e elaboradas, com a finalidade de adaptar a ferramenta aos mais diversos casos de uso. Isso é feito através do desenvolvimento de módulos de código chamados ***Service Providers***; eles implementam interfaces especiais do programa, chamadas ***Service Provider Interfaces*** (SPIs) (Keycloak, 2023). Com esse mecanismo, praticamente todo o comportamento do sistema pode ser alterado.

Por conta de sua flexibilidade e por ter uma comunidade ativa e atualizações constantes, o *Keycloak* foi capaz de se consolidar. Hoje, ele é um dos *softwares* mais proeminentes em seu nicho, sendo considerado uma ótima solução para serviços ligados à identificação eletrônica. O *Keycloak* é utilizado por grandes nomes do mercado nacional e internacional (PAIVA, 2018; Keycloak, 2022).

3 TRABALHOS CORRELATOS

Para definir o escopo do presente trabalho, foi realizado um levantamento das soluções existentes no mercado voltadas para a implementação do protocolo *OpenID Connect for Identity Assurance* 1.0. A equipe responsável pelo desenvolvimento da especificação mantém um repositório que lista os serviços que a incorporam (LODDERSTEDT; SANZ; HAINE, 2023). Essa relação destaca a escassez de trabalhos relacionados ao protocolo estudado, mencionando apenas quatro implementações de *software* e dois serviços em produção que afirmam seguir o protocolo, além de alguns sistemas considerando sua adoção.A falta de implementações torna-se ainda mais notória quando comparada a outros protocolos relacionados à identidade eletrônica e garantia de informações, como o *Financial-grade API* (FAPI) (SAKIMURA; BRADLEY; JAY, 2021), por exemplo. Para efeito de comparação, a página de certificações da *OpenID Foundation* lista centenas de produtos finais, com implementações verificadas e certificadas para o protocolo FAPI, contrastando com apenas duas para o *OpenID Connect for Identity Assurance* 1.0 (OpenID Foundation, 2023).

Por não se relacionarem diretamente com o escopo do projeto, excluíram-se da pesquisa realizada os produtos finais e sistemas de identidade eletrônica que fazem uso do protocolo. Dessa forma, o enfoque do estudo foram bibliotecas, ferramentas para desenvolvedores e plataformas de gerenciamento de acesso e identidade, como o *Keycloak*, que se propunham a implementá-lo. Nas seções seguintes, cada uma das soluções encontradas será detalhada.

3.1 AUTHLETE

O *Authlete* é um serviço baseado em nuvem de identificação e autenticação de usuários. O sistema é produto de uma empresa de mesmo nome, com sede no Japão, e que conta com diversas filiais ao redor do globo (Authlete, 2023a). Dentre usuários do serviço, destacam-se grandes nomes nacionais e internacionais, como o sistema de *internet banking* brasileiro, *Nubank*, e a empresa belga de entretenimento, *DPG Media*.

O sistema foi projetado com o intuito de simplificar o desenvolvimento de aplicações que desejam integrar os protocolos *OAuth* 2.0 e *OpenID Connect* 1.0 em seu funcionamento. Isso é feito através da disponibilização de *Application Programming Interfaces* (APIs)[1] que prometem desempenhar o papel de um servidor de autorização, como descrito nos protocolos citados, sem a necessidade de que se mantenha um banco de dados dedicado para isso. Seus serviços são certificados pela *OpenID Foundation* e garantem suporte nativo a especificações voltadas para a segurança em ambientes financeiros e bancários, como a FAPI (OpenID Foundation, 2023; SAKIMURA; BRADLEY; JAY, 2021).

O sistema da *Authlete* opera em um modelo pago, com planos direcionados ao mercado empresarial (Authlete, 2023b). Entretanto, a empresa disponibiliza, em regime de código-

[1] Uma API é um conjunto de regras e protocolos que permite a interação entre diferentes softwares (FERRAIOLO; LYNCH; TOTH, 1993).

aberto, um conjunto de bibliotecas escritas em Java, com o intuito de possibilitar a conexão de aplicações à plataforma. Dentre elas, a biblioteca `authlete-java-common`[2] conta com uma série de funções que visam automatizar processos ligados à manipulação de dados e atributos, no contexto de identidades eletrônicas tratadas nos protocolos mencionados (Authlete, 2023c).

A plataforma *Authlete* alega conformidade com a especificação *OpenID Connect for Identity Assurance* 1.0 e, inclusive, está listada em seu repositório de desenvolvimento (LODDERSTEDT; SANZ; HAINE, 2023). Além disso, sua biblioteca `authlete-java-common`, a partir da versão 2.2, implementa funções voltadas para facilitar a implantação do protocolo em sistemas diversos. O código é disponibilizado para análise e seu uso é flexível, adequando-se a múltiplos casos de uso (Authlete, 2023c).

Embora o serviço de gerenciamento de identidades disponibilizado pela *Authlete* seja similar ao *Keycloak*, seus propósitos finais se divergem (Authlete, 2023a; Keycloak, 2013). A *Authlete* tem foco em permitir que desenvolvedores adaptem suas próprias aplicações para se conectar à plataforma, enquanto o *Keycloak* preocupa-se com oferecer um sistema completo de identificação eletrônica, em uma ferramenta sem a necessidade de programação. Outra grande diferença diz respeito à política paga, adotada pela *Authlete*, que contrasta com a natureza *open source* do *Keycloak*.

3.2 CONNECT2ID SERVER E NIMBUS

Connect2id Server é o nome de um produto digital sob gestão da organização *Connect2id*, uma empresa de tecnologia búlgara (Connect2id, 2023a). Lançado em fevereiro de 2014, seu objetivo é permitir o uso de protocolos relacionados ao *OpenId Connect* 1.0 e *OAuth* 2.0 em aplicações diversas (Connect2id, 2023b). Para isso, o *Connect2id Server* utiliza mecanismos similares aos adotados pelo *Authlete*: uma série de *web* APIs são disponibilizadas com o intuito de permitir a conexão ao sistema. Essa estratégia permite que os desenvolvedores que optem por utilizar a plataforma possam ter maior controle da ferramenta e adaptabilidade a seus casos de uso específicos. Toda a integração pode ser feita utilizando suas próprias políticas, *interfaces* ou serviços, por exemplo.

O *Connect2id Server* também possui certificação provida pela *OpenID Foundation* e conta com suporte nativo a soluções voltadas para identificação eletrônica em áreas de saúde, governo e finanças (OpenID Foundation, 2023; Connect2id, 2023b). Assim como o *Authlete*, seu foco é empresarial e seus serviços são pagos, mas há a opção de testar a plataforma gratuitamente (Connect2id, 2023d). A empresa também disponibiliza bibliotecas *open source*, escritas em Java, para desenvolvimento de clientes e servidores que sigam os protocolos *OAuth* 2.0 e *OpenID Connect* 1.0: a família de bibliotecas *Nimbus*.

Dentre as bibliotecas sob essa família, destaca-se a `oauth2-oidc-sdk`[3] (Connect2id,

[2] O código-fonte da biblioteca `authlete-java-common` pode ser encontrado através do endereço apontado pelo *link*: **https://github.com/authlete/authlete-java-common**.

[3] Mais informações acerca da biblioteca `oauth2-oidc-sdk` e seu funcionamento podem ser acessadas por meio da URL: **https://connect2id.com/products/nimbus-oauth-openid-connect-sdk**.

2023c). Ela permite e facilita o desenvolvimento de aplicações compatíveis com as especificações de identificação discutidas, além de fornecer ferramentas para manipulação de dados nesse contexto. Ambos os produtos *Connect2id Server* e as bibliotecas *Nimbus* garantem suporte ao *OpenID Connect for Identity Assurance* 1.0 e estão listados em seu repositório oficial (LODDERSTEDT; SANZ; HAINE, 2023). Assim como o *Authlete*, o *Connect2id Server* diverge dos propósitos do *Keycloak*, por motivos semelhantes.

3.3 EKYC HUB

Esse é o único projeto mencionado na listagem de implementações oficiais do *OpenID Connect for Identity Assurance* 1.0 que se propõe a trazer o protocolo para o *Keycloak* (LODDERSTEDT; SANZ; HAINE, 2023). O eKYC *Hub* promete ser um *framework* que implementa a especificação estudada e permite a conexão do *Keycloak* com provedores de dados verificados (Identity First Tech, 2021). Ou seja, o objetivo desse projeto é utilizar a infraestrutura do *Keycloak* para receber requisições de atributos verificados, processá-las, consumir os atributos requisitados de um servidor externo e entregá-los para aplicações clientes.

O repositório de desenvolvimento do eKYC *Hub*[4], na plataforma *GitHub*, é sua única fonte de documentação. Nesse repositório, a autoria do projeto é reclamada por um grupo chamado *Identity First Tech*. A última atualização da solução data de dezembro de 2021. Ademais, comentários deixados no repositório, antes dessa data, alegando problemas na demonstração do produto, não obtiveram resposta. Outro ponto que merece destaque é o fato de que, na data de produção do presente trabalho, os *links* disponibilizados com detalhes da solução e informações sobre o grupo de desenvolvimento não funcionam ou apontam para páginas desatualizadas.

Em sua documentação, menciona-se que o eKYC *Hub* foi idealizado para funcionar em conjunto com um serviço de verificação de identidades eletrônicas chamado *Passbase*. No entanto, informações acerca desse produto não puderam ser recuperadas. Os *links* que deveriam redirecionar o usuário para a página desse serviço apontam para o destino incorreto e pesquisas sobre o sistema, na *internet*, não retornam resultados relevantes.

Esses desafios, combinados com problemas no código-fonte da demonstração fornecida pelo grupo de desenvolvimento, como evidenciado nos comentários supracitados, dificultam a avaliação desta solução e desencorajam sua adoção para dar suporte às funcionalidades do protocolo no *Keycloak* em serviços e aplicações do mundo real. Adicionalmente, apesar da proposta do eKYC *Hub* em ser uma solução modular, a necessidade de depender exclusivamente de uma fonte externa para a recuperação de dados verificados representa uma desvantagem nesse aspecto, já que essa abordagem exige a implementação (ou contratação) de um servidor de verificação de identidades.

[4] O repositório de desenvolvimento do eKYC *Hub*, contendo mais informações, pode ser acessado através do seguinte endereço: **https://github.com/identityfirst/eKYC-Hub**.

3.4 OUTRAS IMPLEMENTAÇÕES

Além das implementações listadas oficialmente pelo repositório de desenvolvimento do *OpenID Connect for Identity Assurance* 1.0, uma busca por soluções relacionadas ao tópico de estudo do presente trabalho foi feita em fóruns de desenvolvimento e plataformas de compartilhamento de código, como o *GitHub*. O objetivo desse levantamento foi mapear projetos que se propusessem a implementar o protocolo estudado e ainda estivessem em desenvolvimento, ou que ainda não tivessem sido oficialmente incorporadas ao repositório. O foco dessa pesquisa foi especificamente direcionado a soluções que envolvessem o *Keycloak*, com a finalidade de analisar formas de implementação distintas das apresentadas anteriormente.

Durante essa busca, uma única solução relevante foi encontrada, nos fóruns de discussão do repositório de desenvolvimento do *Keycloak* no *GitHub*[5]. Essa solução foi proposta em junho de 2023 e promete trazer funcionalidades nativas do *OpenID Connect for Identity Assurance* 1.0 para o sistema. A finalidade do projeto é modificar o código-fonte do *Keycloak* para que requisições e repostas pertencentes ao protocolo sejam reconhecidas e processadas. Para isso, essa implementação se propõe a possibilitar a comunicação entre o sistema e serviços de verificação de identidade, por meio de requisições HTTP, de maneira semelhante ao eKYC *Hub* (Identity First Tech, 2021).

Apesar dos esforços do projeto em implementar o protocolo no ambiente do *Keycloak* serem válidos, uma série de problemas foram encontrados durante sua análise e divergências em relação ao protocolo estudado foram identificadas. Durante investigação da solução, constatou-se que o sistema é incapaz de omitir corretamente elementos de repostas à requisições do protocolo, conforme descrito na Seção 6.5 da especificação. Outro problema experienciado foi a presença de erros bloqueantes durante o processamento de requisições. A solução não permitiu a requisição de `verified_claims` com múltiplos elementos, por exemplo, como descrito na Seção 6.4 do protocolo. Algumas funcionalidades descritas pela especificação, como a possibilidade de requisitar *claims* como essenciais, não foram implementadas. Ademais, oportunidades de melhoria da solução foram constatadas. Essa implementação, por exemplo, não realiza a validação dos objetos JSON recebidos e, portanto, faz (ou tenta fazer) todo o processamento de requisições, mesmo que inválidas.

Adicionalmente, a estratégia de sugerir alterações no código-fonte do *Keycloak* apresenta uma série de desvantagens adicionais a essa implementação: para que a solução seja implementada com êxito, é necessário que a equipe de desenvolvimento do *Keycloak* concorde em avaliá-la e possivelmente incorporá-la, um processo demorado que pode não se concretizar; as aplicações que optarem por adotar a solução, caso ela não seja oficialmente aceita, precisarão depender do código modificado do *Keycloak*, copiando todo o seu repositório e realizando o processo de compilação manualmente, um procedimento complexo, moroso e suscetível a erros; além disso, o projeto ficará vinculado a uma versão específica do *Keycloak*, tornando

[5] A solução pode ser encontrada através do *link*: **https://github.com/keycloak/keycloak/pull/21309**

impossível a realização de atualizações, já que o código-fonte foi alterado para dar suporte ao protocolo, exigindo uma reconstrução completa do trabalho caso o código seja modificado para uma nova versão.

3.5 CONSIDERAÇÕES E RESULTADO DA ANÁLISE

Até a data de produção do presente trabalho, não há, por parte do time de desenvolvimento do *Keycloak*, um esforço para implementação do *OpenID Connect for Identity Assurance* 1.0 de maneira nativa, assim como feito em outros sistemas de gestão de identidades eletrônicas. Com isso em mente, aplicações que optem por utilizar o protocolo estudado precisam recorrer a soluções de organizações terceiras, como as listadas nas seções anteriores, para atendê-lo.

Nesse sentido, embora as implementações apresentadas pelas plataformas *Authlete* e *Connect2id Server* apresentem soluções concretas e bem estabelecidas, seus propósitos de uso divergem da finalidade de utilização do *Keycloak*. Ademais, suas implementações caracterizam-se como produtos pagos, o que não só dificulta a democratização e popularização do protocolo, mas também foge ao escopo desse projeto.

No contexto de soluções gratuitas voltadas especificamente ao *Keycloak*, a pesquisa realizada evidencia que nenhuma das implementações disponíveis no mercado apresentam-se como alternativas factíveis para implantação do protocolo. O eKYC *Hub*, apesar de ser reconhecido como uma das soluções oficiais pela equipe responsável de desenvolvimento do protocolo, apresenta problemas em sua implementação, possui escassez de documentação e carece de manutenção. Dessa forma, o projeto desenvolvido pelo grupo *Identity First Tech* não se caracteriza como uma opção viável para adoção do *OpenID Connect for Identity Assurance* 1.0 em sistemas reais. Já a implementação do protocolo não listada oficialmente no repositório, descrita na Seção 3.4, apresenta uma série de falhas e limitações, se comparada à especificação. Além disso, essa solução tem a desvantagem de estar ligada à modificação do código-fonte do *Keycloak* para seu funcionamento, o que traz uma série de impedimentos. Esses fatores prejudicam adoção dessa solução por sistemas que estejam dispostos a utilizar o protocolo, e inviabilizam sua possibilidade de uso.

Tendo em vista a pesquisa realizada, portanto, conclui-se que nenhuma das soluções apresentadas cumpre, de forma satisfatória, com os requisitos definidos no escopo desse projeto: trazer suporte ao protocolo *OpenID Connect for Identity Assurance* 1.0 em uma ferramenta gratuita, de maneira modular e simples. Nesse aspecto, o presente trabalho justifica-se, pretendendo trazer avanços significativos em relação à adoção e popularização do protocolo estudado.

Por fim, convém a menção de que as bibliotecas em Java, voltadas à integração com sistemas de identificação eletrônica mencionadas nas seções anteriores, embora tenham sido projetadas para conectar aplicações a plataformas específicas, podem ser adaptadas para utilização em projetos mais amplos. Essa abordagem facilita a implementação do *OpenID Connect for Identity Assurance* 1.0 em sistemas que não o suportam, mas ainda assim exige que um esforço significativo seja despendido para que essa implementação seja materializada.

4 DESENVOLVIMENTO

Após análise e levantamento de recursos, foi possível dar inicio ao estágio de desenvolvimento de uma solução própria. Neste capítulo, serão abordados os detalhes desse desenvolvimento e o processo utilizado para materialização do projeto.

4.1 PREPARAÇÃO DO AMBIENTE

Para iniciar o processo de construção da solução, foi necessário primeiramente que o ambiente de desenvolvimento fosse configurado. A tarefa inicial dessa preparação foi configurar uma instância do *Keycloak*, já que a implementação seria projetada para funcionar em conjunto com esse sistema de gestão de identidades. Esse processo foi feito com o auxílio da utilização de contêineres *Docker*.

A plataforma *Docker* é um projeto de código aberto que simplifica a criação, implementação e execução de aplicativos em um sistema de contêineres (Docker, 2023a). Esses contêineres são máquinas virtuais *lightweight* que encapsulam aplicações e incluem todas as dependências necessárias para sua execução, como bibliotecas, códigos e configurações (ANDERSON, 2015). As máquinas virtuais podem ser configuradas do zero, mas organizações costumam disponibilizar contêineres já configurados com suas aplicações, através do uso de imagens *Docker*. Nesse contexto, são disponibilizas para uso, pelo time do *Keycloak*, diversas dessas imagens, contendo cada uma de suas versões. Para o desenvolvimento do presente trabalho, foi utilizada uma das imagens correspondentes ao *Keycloak*, em sua versão `22.0.5`.

Adicionalmente ao *Keycloak*, outra ferramenta foi utilizada: o `keycloak-config-cli` (Adorsys GmbH & Co. KG, 2023). Essa aplicação serve ao propósito de configurar instâncias do *Keycloak*, a partir de arquivos de configuração, escritos em JSON. Essa funcionalidade é especialmente útil para que as configurações feitas não sejam perdidas durante reinicializações e recompilações do código. O `keycloak-config-cli` também está disponível através de uma imagem *Docker* e sua versão `5.9.0-22.0.4` foi utilizada.

Para que fosse possível lidar com múltiplos contêineres simultaneamente, utilizou-se a ferramenta *Docker Compose* (Docker, 2023b). Um recurso que permite o uso e gestão de diversas máquinas virtuais *Docker* através de um único arquivo de configuração.

Com uma instância do *Keycloak* devidamente configurada e pronta para uso, deu-se início ao processo de preparação do ambiente de criação do módulo que seria responsável por adicionar as funcionalidades do protocolo estudado ao sistema. Para que a solução fosse verdadeiramente modular, optou-se pelo desenvolvimento de um *Service Provider* personalizado. *Service Providers*, no *Keycloak*, devem seguir uma estrutura pré-definida e cumprir com alguns requisitos para que sejam reconhecidos pelo sistema (Keycloak, 2023).

Além de implementarem as *Service Provider Interfaces*, as classes correspondentes a

um *Service Provider* devem estar encapsuladas em um arquivo do tipo *Java Archive* (JAR)[1]. Para que isso fosse feito, optou-se pela implementação de um projeto *Maven*, para o presente trabalho. O *Maven* é uma ferramenta, sob gestão do grupo *Apache Software Foundation*, voltada especialmente a gerenciamento de projetos (The Apache Software Foundation, 2023). Seu objetivo é simplificar o processo de construção, execução e gestão de dependências e um de seus diferenciais é seu repositório centralizado que simplifica a inclusão e atualização de bibliotecas e componentes externos em um projeto (VARANASI, 2019).

Adicionalmente, para que um *Service Provider* personalizado seja reconhecido por uma instância do *Keycloak*, o arquivo JAR contendo seu código-fonte deve ser adicionado à um diretório específico. Esse arquivo precisa ser transferido para a pasta `providers` dentro do ambiente do *Keycloak*. Para que isso fosse feito, utilizou-se um arquivo especial de configuração de contêineres *Docker*, chamado *Dockerfile* (Docker, 2023c).

Por fim, um pequeno *shell script* foi escrito, para automatizar o processo de compilação e execução do projeto. Um *shell script* é um arquivo de texto contendo uma sequência de comandos destinados a serem executados em um *shell*, que é a interface de linha de comando de um sistema operacional (KERNIGHAN; MASHEY, 1979).

4.2 DESENVOLVIMENTO DA SOLUÇÃO

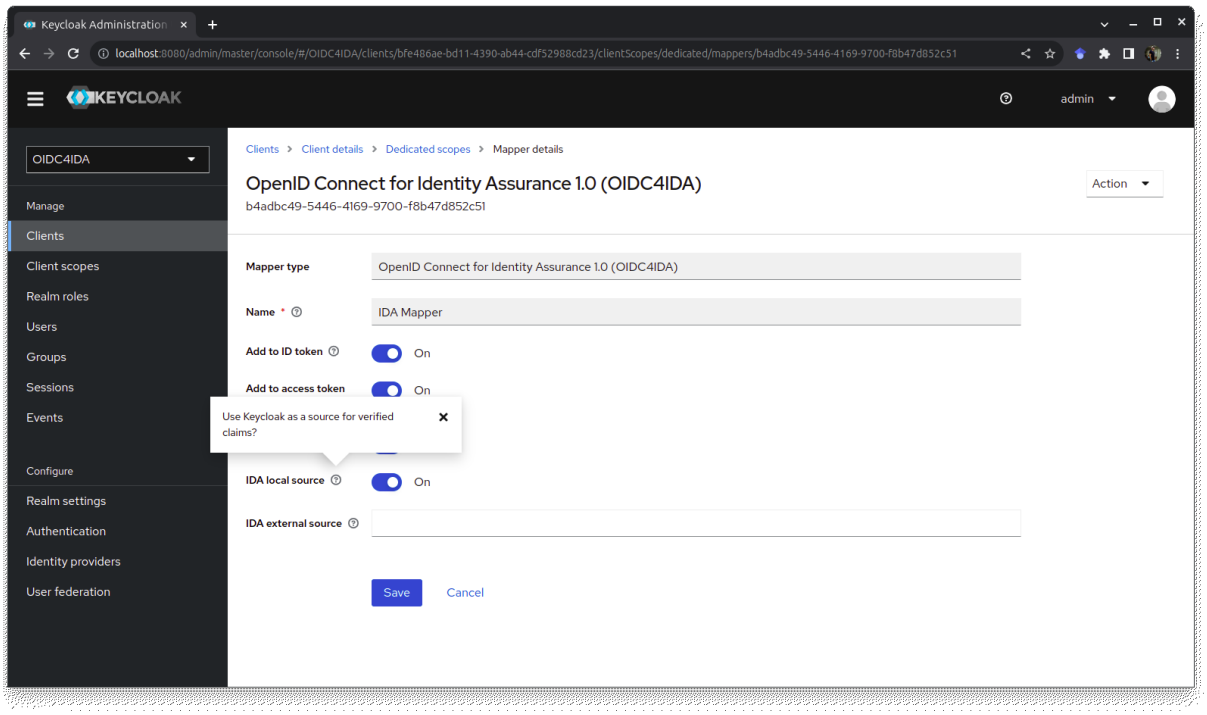
Após a configuração do ambiente de desenvolvimento, descrita na seção anterior, pôde se dar início a implementação da solução. Para esse momento inicial, utilizou-se a implementação do *OpenID Connect for Identity Assurance* 1.0, apresentada na Seção 3.4, como base para o projeto. Como a proposta do trabalho realizado era o desenvolvimento de uma aplicação modular, deu-se início à extração do código modificado pela solução base, de dentro do código-fonte do *Keycloak*. Esse processo demandou uma etapa de estudo, identificação e adaptação do código, para que fosse concluído de forma satisfatória.

Uma vez que a solução foi extraída e adaptada, foi possível utilizar sua organização para nortear a estruturação do projeto. Essa estratégia foi particularmente útil para que fossem identificadas quais *Service Provider Interfaces* deveriam ser implementadas, bem como quais classes e métodos precisavam ser utilizados. Visando trazer mais modularidade e independência ao projeto, a opção de consumir dados verificados diretamente do banco de dados do *Keycloak* foi introduzida. Como uma funcionalidade extra aos objetivos do trabalho, decidiu-se, também, manter a oportunidade de que os atributos verificados fossem consumidos através de um servidor de verificação externo, por meio de requisições HTTP. A escolha de qual fonte de informações deve ser utilizada foi construída de modo que isso possa ser alterado nas configurações do *Keycloak*, por meio da *interface* de administração do sistema. A Figura 5 demonstra a tela de configuração e seus atributos.

Quanto à lógica de processamento de repostas e requisições referentes ao protocolo

[1] Arquivos JAR são utilizados para agregar múltiplos arquivos em um único arquivo executável (Oracle, 2023).

Figura 5 – Tela de configuração do *OpenID Connect for Identity Assurace* 1.0 no *Keycloak*.



Fonte: O autor.

estudado na solução tomada como base, toda a sua implementação precisou ser refeita, por conta de erros e inconsistências em relação ao protocolo. Seguindo essa abordagem, foi possível garantir que as especificações do protocolo estavam sendo seguidas em sua completude e que melhorias fossem feitas na solução. Dentre as melhorias mais significativas, destaca-se a implementação de validação dos objetos JSONs manipulados, através de JSON *schemas*.

JSON *schemas* são esquemas que definem a estrutura, o formato e as restrições de dados em objetos JSON (WRIGHT et al., 2022). Esses esquemas são especialmente úteis para garantir a consistência e a validade de dados em objetos JSON de larga escala, como os lidados pelo *OpenID Connect for Identity Assurance* 1.0 (WRIGHT; ANDREWS; HUTTON, 2022). A própria equipe de desenvolvimento do protocolo disponibiliza, em seu repositório oficial, JSON *schemas* para validação de dados no contexto da especificação, que foram utilizados na solução proposta pelo presente trabalho.

Finalmente, para que fosse implementado o processamento de requisições, utilizaram-se as ferramentas providas pela biblioteca `authlete-java-common`, apresentada na Seção 3.1. Essa decisão trouxe não só agilidade na implementação da solução, como também maior segurança e confiabilidade ao seu funcionamento, que passou a utilizar ferramentas reconhecidas pelo grupo de desenvolvimento do protocolo estudado.

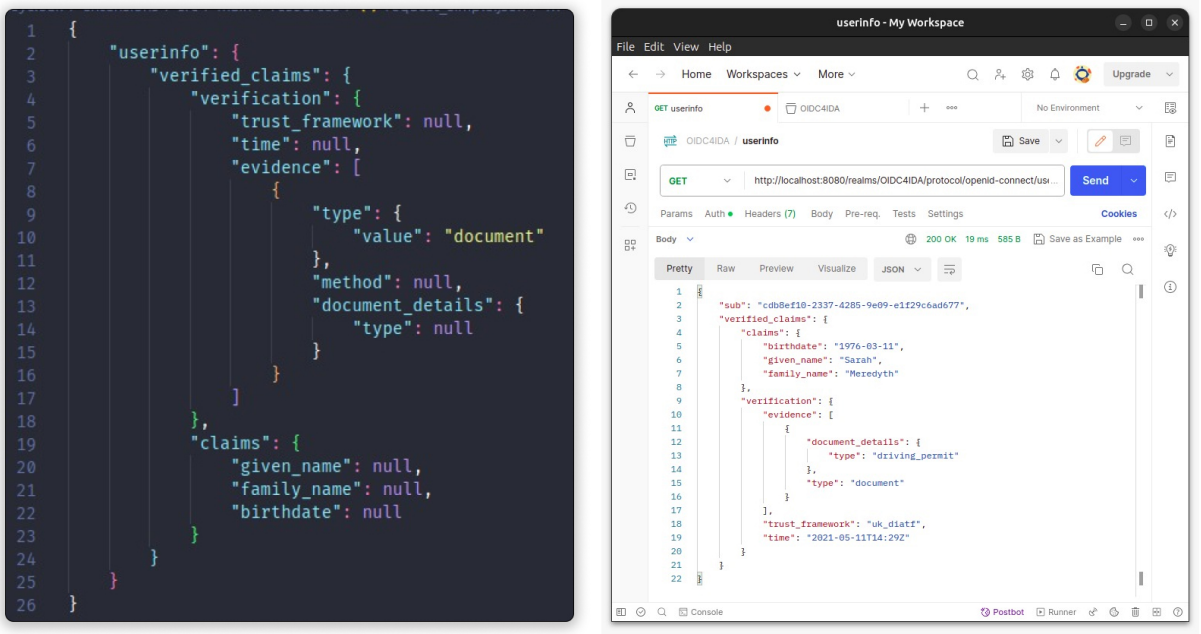
5 RESULTADOS

Após a materialização da solução proposta, uma série de testes foram conduzidos, através da ferramenta *Postman*: uma plataforma para uso e desenvolvimento de APIs, que permite a simulação de requisições HTTP (Postman, Inc., 2023). Além de oferecer uma interface gráfica otimizada para visualização e interpretação de resultados, o *Postman* apresenta integração com serviços *web*, incluindo suporte aos protocolos *OAuth* 2.0 e *OpenID Connect* 1.0.

Para conduzir a bateria de testes, foram utilizados os arquivos de exemplo disponibilizados no repositório de desenvolvimento do *OpenID Connect for Identity Assurance* 1.0[1]. Um ambiente de simulação também precisou ser configurado no *Keycloak*, através da ferramenta `keycloak-config-cli`. Buscando apresentar um pouco do processo de validação da solução proposta, alguns exemplos de repostas e requisições que corroboram com a homologação da implementação são oferecidos. A Figura 6 ilustra um exemplo simples de requisição e resposta de atributos verificados validados por meio da apresentação de documentos. Este teste valida a capacidade da solução de filtrar elementos conforme necessário e quando solicitado.

Figura 6 – Exemplo de requisição descrita na Seção 7.1 da documentação do *OpenID Connect for Identity Assurance* 1.0 (LODDERSTEDT et al., 2022).

`Requisição` `Resposta`



Fonte: O autor.

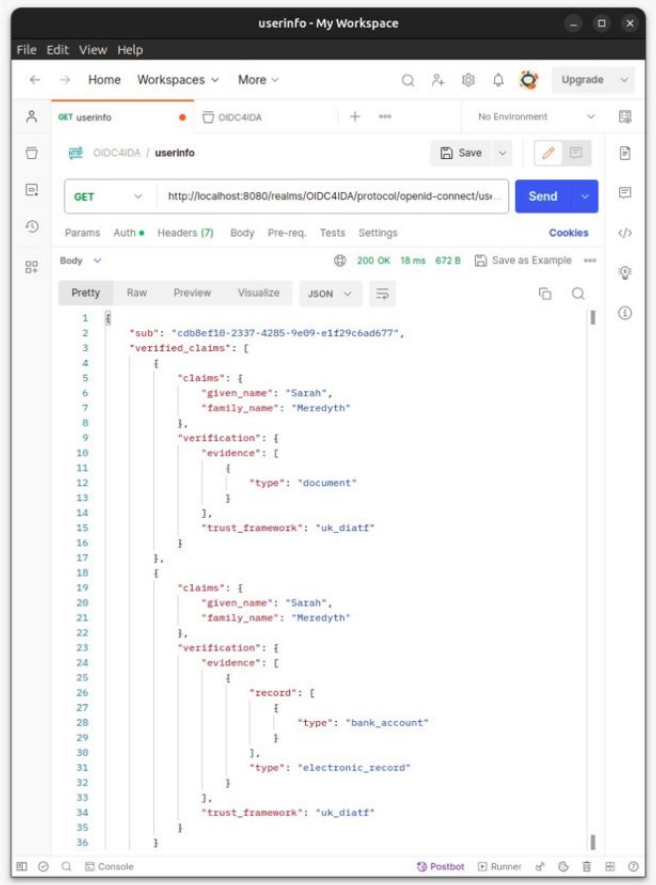
[1] O arquivo contendo a coleção de atributos utilizados nos testes está disponível em: **https://shorturl.at/dCI49**.

Figura 7 – Exemplo de requisição múltipla, como descrito na Seção 6.4 da documentação do *OpenID Connect for Identity Assurance* 1.0 (LODDERSTEDT et al., 2022).

Requisição

```
{
    "userinfo": {
        "verified_claims": [
            {
                "verification": {
                    "trust_framework": { "value": "uk_diatf" },
                    "evidence": [ {
                        "type": { "value": "document" }
                    } ]
                },
                "claims": {
                    "given_name": null,
                    "family_name": null
                }
            },
            {
                "verification": {
                    "trust_framework": { "value": "uk_diatf" },
                    "evidence": [ {
                        "type": { "value": "electronic_record" },
                        "record": [ {
                            "type": { "value": "bank_account" }
                        } ]
                    } ]
                },
                "claims": {
                    "given_name": null,
                    "family_name": null
                }
            }
        ]
    }
}
```

Resposta



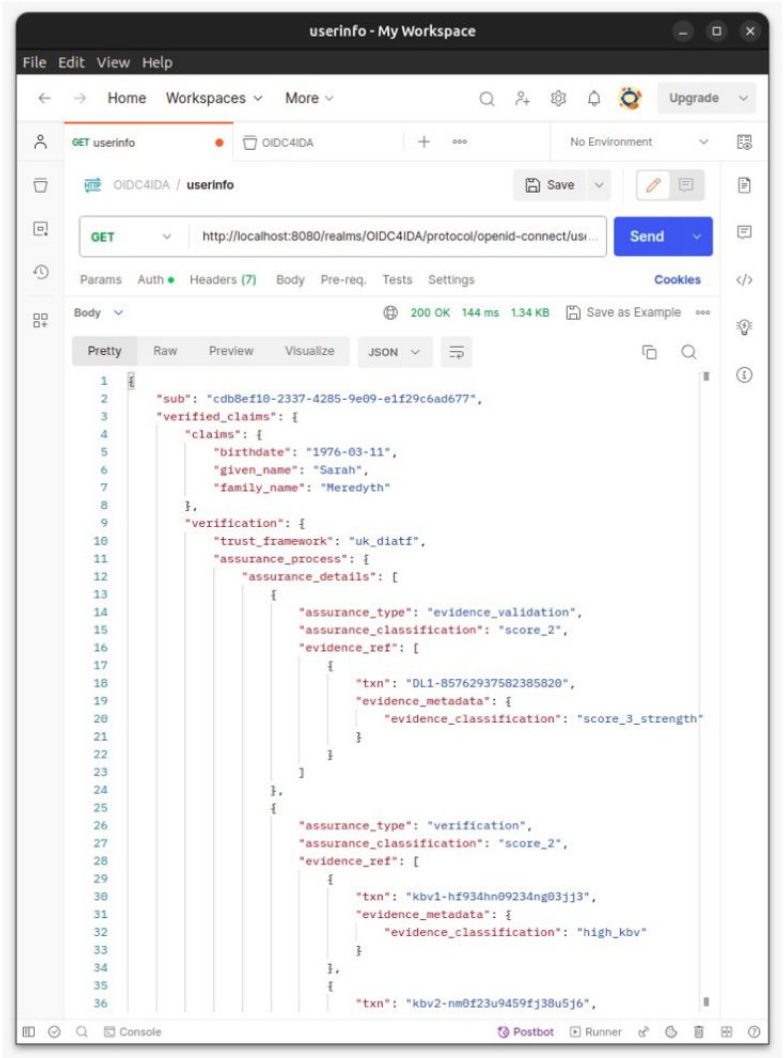
Fonte: O autor.

Figura 8 – Exemplo de requisição com `assurance_details`, como descrito na Seção 6.2 da documentação do *OpenID Connect for Identity Assurance* 1.0 (LODDERSTEDT et al., 2022).

Requisição

```
{
    "userinfo": {
        "verified_claims": {
            "verification": {
                "trust_framework": null,
                "assurance_process": {
                    "assurance_details": [ {
                        "assurance_type": { "value": "verification" }
                    } ]
                }
            },
            "claims": {
                "given_name": null,
                "family_name": null,
                "birthdate": null
            }
        }
    }
}
```

Resposta



Fonte: O autor.

A Figura 7 destaca a possibilidade de solicitar vários artefatos de verificação, evidenciando a capacidade da implementação de lidar com requisitos e requisições diversas. Por fim, a Figura 8 confirma que até mesmo os detalhes mínimos da especificação foram atendidos. Nesse exemplo, é demonstrado o processamento de uma requisição que inclui o objeto `assurance_details`, com solicitações específicas para filtragem. Contudo, a solução opta por ignorar o pedido de filtragem, conforme descrito na Seção 6.2 da especificação do protocolo, entregando o objeto `assurance_details` na sua totalidade.

A etapa de testes validou a solução proposta e permitiu que a identificação e correção de *bugs* e mau funcionamentos fosse realizada. Com esse desfecho, o projeto de implementação pôde ser considerado como concluído, nos requisitos propostos inicialmente. A solução desenvolvida, em conjunto com mais informações técnicas acerca do projeto, pode ser encontrada no repositório do autor, disponível através do endereço: **https://github.com/Bredstone/Keycloak-Extension-OIDC4IDA**.

6 CONCLUSÕES

Ao longo do desenvolvimento do presente trabalho, um abrangente levantamento de implementações do protocolo *OpenID Connect for Identity Assurance* 1.0 pôde ser realizado. Esse mapeamento permitiu a análise das características, pontos fortes e limitações das soluções encontradas. O estudo proporcionou, ainda, maior compreensão acerca da especificação examinada, viabilizando que fosse tomado conhecimento não só de suas peculiaridades técnicas, como também dos desafios para sua efetiva adoção e popularização. Em um caráter ainda mais relevante, a pesquisa realizada possibilitou a identificação de uma lacuna no mercado, referente a implementação do protocolo estudado em um dos sistemas gerenciadores de identidades eletrônicas mais proeminentes, o *Keycloak*.

Tendo isso em mente, a proposta de materialização de uma solução que trouxesse suporte ao protocolo no *Keycloak* pôde ser feita. Essa solução, cujo processo de desenvolvimento foi detalhado ao longo do presente trabalho, demonstrou-se rica ao não apenas atender aos requisitos definidos no início do projeto, como também trazer funcionalidades extras às idealizadas. A escolha por desenvolver uma solução integrada ao *Keycloak* destaca a implementação apresentada, por garantir o suporte ao *OpenID Connect for Identity Assurance* 1.0 em uma ferramenta *no code* gratuita, amplamente reconhecida e consolidada no mercado.

Além disso, a proposta de desenvolvimento de uma solução modular traz uma série de vantagens quanto à viabilidade do produto apresentado. Essa abordagem permite que o protocolo possa ser integrado a sistemas já em produção de maneira facilitada, além de viabilizar sua interoperabilidade entre versões diversas do *Keycloak*. A estratégia modular adotada contempla, ainda, a possibilidade de que alterações no código-fonte da solução possam ser realizadas, sem que seja necessária a modificação de arquivos pertencentes ao *core* do *Keycloak*. Essa característica, em conjunto com uma documentação de código detalhada e estruturação clara de projeto, permite que desenvolvedores que optem por utilizar a solução oferecida possam modificá-la para melhor atender aos seus casos de uso específicos, de maneira ágil e simplificada.

Dentre os aspectos mais importantes da solução, destaca-se a utilização da biblioteca `authlete-java-common`. A decisão de sua aplicação no projeto garante a conformidade ao protocolo estudado em mais alto nível de rigorosidade. Por ser listada como referência pelo time de desenvolvimento do protocolo, a ferramenta da *Authlete* traz maior segurança quanto ao processamento de repostas e requisições referentes a especificação e agrega valor à solução proposta. Esse detalhe de implementação, adicionado à validação dos artefatos manipulados pelo sistema por meio da utilização de JSON *schemas*, contribui para que a implementação apresentada destaque-se entre as demais disponibilizadas no mercado.

Ademais, a possibilidade de escolher entre consumir atributos verificados diretamente do banco de dados do *Keycloak* ou de um servidor externo de verificação é uma funcionalidade diferenciada dessa implementação, que não foi mapeada em outras soluções estudadas. Adicionalmente, essa decisão de projeto corrobora com a modularidade da solução, visto que essa

abordagem faz com que um servidor externo de verificação não seja estritamente necessário para o armazenamento de informações.

Por fim, conclui-se que a implementação proposta durante o desenvolvimento do presente projeto contribui positivamente para efetiva adoção e popularização do *OpenID Connect for Identity Assurance* 1.0, oferecendo seu suporte a uma plataforma de relevância no contexto computacional. Por estar associada a uma ferramenta *open source* e *no code*, a solução apresentada colabora para a democratização do protocolo, promovendo um ambiente mais seguro no cenário digital.

6.1 TRABALHOS FUTUROS

Embora os requisitos de projeto tenham sido atingidos durante o desenvolvimento do presente estudo, possibilidades de melhoria e enriquecimento da solução foram identificados nesse processo. Dentre elas, o amadurecimento do código, através da implementação de testes unitários automatizados e a divulgação do modelo produzido na comunidade de desenvolvimento do *Keycloak* e no repositório oficial do *OpenID Connect for Identity Assurance* 1.0.

Adicionalmente, estuda-se a possibilidade de implantação da solução proposta nesse trabalho em um sistema de identificação eletrônica real, com abrangência nacional, a Identificação Eletrônica do Registro Civil do Brasil (IdRC) (SILVA et al., 2023). Desse modo, espera-se a consolidação da implementação e fomento à utilização do protocolo.

REFERÊNCIAS

ABBATE, J. Government, business, and the making of the internet. **Business History Review**, 2001.

Adorsys GmbH & Co. KG. **keycloak-config-cli**. 2023. Disponível em: https://github.com/adorsys/keycloak-config-cli.

ANDERSON, C. Docker [software engineering]. **Ieee Software**, IEEE, v. 32, n. 3, p. 102–c3, 2015.

ARGYRIOU, M.; DRAGONI, N.; SPOGNARDI, A. Security flows in oauth 2.0 framework: A case study. In: . [S.l.: s.n.], 2017. p. 396–406. ISBN 978-3-319-66283-1.

Authlete. **Authlete - Homepage**. 2023. Disponível em: https://www.authlete.com/.

Authlete. **Authlete - Pricing**. 2023. Disponível em: https://www.authlete.com/pricing/.

Authlete. **Authlete Common Library for Java**. 2023. Disponível em: https://github.com/authlete/authlete-java-common/tree/master.

Brasil. Lei nº 13.709, de 14 de agosto de 2018. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm.

BRAY, T. **RFC 8259: The JavaScript object notation (JSON) data interchange format**. [S.l.]: RFC Editor, 2017.

BROSS, B. Theseus' paradox:: History, authenticity and identity. **ARCC Conference Repository**, v. 1, n. 1, May 2019. Disponível em: http://arcc-repository.org/index.php/repository/article/view/638.

CLERCQ, J. D. Single sign-on architectures. In: SPRINGER. **Infrastructure Security: International Conference, InfraSec 2002 Bristol, UK, October 1–3, 2002 Proceedings**. [S.l.], 2002. p. 40–58.

Connect2id. **About Connect2id**. 2023. Disponível em: https://connect2id.com/about.

Connect2id. **Connect2id server**. 2023. Disponível em: https://connect2id.com/products/server.

Connect2id. **Nimbus OAuth 2.0 SDK with OpenID Connect extensions**. 2023. Disponível em: https://connect2id.com/products/nimbus-oauth-openid-connect-sdk.

Connect2id. **Online demo / Downloads**. 2023. Disponível em: https://connect2id.com/products/server/download.

DENNISS, W.; BRADLEY, J. **OAuth 2.0 for Native Apps**. RFC Editor, 2017. RFC 8252. (Request for Comments, 8252). Disponível em: https://www.rfc-editor.org/info/rfc8252.

DIVYABHARATHI, D.; CHOLLI, N. G. A review on identity and access management server (keycloak). **International Journal of Security and Privacy in Pervasive Computing (IJSPPC)**, IGI Global, v. 12, n. 3, p. 46–53, 2020.

Docker. **Docker - Homepage**. 2023. Disponível em: https://www.docker.com/.

Docker. **Docker Compose overview**. 2023. Disponível em: https://docs.docker.com/compose/.

Docker. **Dockerfile reference**. 2023. Disponível em: https://docs.docker.com/engine/reference/builder/.

DOMENECH, M. C.; COMUNELLO, E.; WANGHAM, M. S. Identity management in e-health: A case study of web of things application using openid connect. In: IEEE. **2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom)**. [S.l.], 2014. p. 219–224.

European Commission. **Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)**. European Commission, 2016. Disponível em: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

European Parliament and Council of the European Union. **Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market**. 2014. Disponível em: https://eur-lex.europa.eu/eli/reg/2014/910/oj.

FERDOUS, M. S.; NORMAN, G.; POET, R. Mathematical modelling of identity, identity management and other related topics. In: **Proceedings of the 7th International Conference on Security of Information and Networks**. [S.l.: s.n.], 2014. p. 9–16.

FERRAIOLO, D. F.; LYNCH, N.; TOTH, P. R. Minimum security requirements for multi-user operating systems. David F. Ferraiolo, N Lynch, Patricia R. Toth, 1993.

FETT, D.; KÜSTERS, R.; SCHMITZ, G. A comprehensive formal security analysis of oauth 2.0. In: **Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security**. [S.l.: s.n.], 2016. p. 1204–1215.

FIELDING, R.; NOTTINGHAM, M.; RESCHKE, J. **RFC 9110: HTTP Semantics**. [S.l.]: RFC Editor, 2022.

FOTIOU, N. et al. Oauth 2.0 authorization using blockchain-based tokens. **arXiv preprint arXiv:2001.10461**, 2020.

GRASSI, P. et al. **Digital Identity Guidelines - Enrollment and Identity Proofing Requirements**. Gaithersburg, MD, 2017.

GRASSI, P.; GARCIA, M.; FENTON, J. **Digital Identity Guidelines**. Gaithersburg, MD, 2017.

HAMMER-LAHAV, E. **The OAuth 1.0 Protocol**. RFC Editor, 2010. RFC 5849. (Request for Comments, 5849). Disponível em: https://www.rfc-editor.org/info/rfc5849.

HARDT, D. **The OAuth 2.0 Authorization Framework**. RFC Editor, 2012. RFC 6749. (Request for Comments, 6749). Disponível em: https://www.rfc-editor.org/info/rfc6749.

HUNT, P. et al. **Security Event Token (SET)**. RFC Editor, 2018. RFC 8417. (Request for Comments, 8417). Disponível em: https://www.rfc-editor.org/info/rfc8417.

Identity First Tech. **eKYC Hub**. 2021. Disponível em: https://github.com/identityfirst/eKYC-Hub.

ISO Central Secretary. **IT Security and Privacy—A Framework for Identity Management-Part 1: Terminology and Concepts**. [S.l.]: International Organization for Standardization, 2019.

ISO, I. **IEC 29115: 2013—Information Technology—Security Techniques—Entity Authentication Assurance Framework**. 2013.

JONES, M. B.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. RFC Editor, 2015. RFC 7519. (Request for Comments, 7519). Disponível em: https://www.rfc-editor.org/info/rfc7519.

KERNIGHAN, B. W.; MASHEY, J. R. The unix™ programming environment. **Software: Practice and Experience**, Wiley Online Library, v. 9, n. 1, p. 1–15, 1979.

Keycloak. **Open source identity and access management**. 2013. Disponível em: https://www.keycloak.org/.

Keycloak. **Keycloak Adopters**. GitHub, 2022. Disponível em: https://github.com/keycloak/keycloak/blob/main/ADOPTERS.md.

Keycloak. **Server Developer Guide**. 2023. Disponível em: https://www.keycloak.org/docs/latest/server_development/.

KIENNERT, C.; BOUZEFRANE, S.; THONIEL, P. Authentication systems. In: **Digital identity management**. [S.l.]: Elsevier, 2015. p. 95–135.

LESTERJR., F. K.; LAMBDIN, D. V. The ship of theseus and other metaphors for thinking about what we value in mathematics education research. In: . [S.l.: s.n.], 1998.

LODDERSTEDT, T. et al. **OAuth 2.0 Security Best Current Practice**. [S.l.], 2023. Work in Progress. Disponível em: https://datatracker.ietf.org/doc/draft-ietf-oauth-security-topics/23/.

LODDERSTEDT, T.; FETT, D. **OpenID Connect for Identity Assurance**. OpenID, 2019. Disponível em: https://openid.net/wordpress-content/uploads/2019/03/openid-connect-4-identity-assurance-00.html.

LODDERSTEDT, T. et al. **OpenID Connect for Identity Assurance 1.0**. OpenID, 2022. Disponível em: https://openid.net/specs/openid-connect-4-identity-assurance-1_0.html.

LODDERSTEDT, T.; SANZ, M.; HAINE, M. **OpenID Connect for Identity Assurance Implementations**. 2023. Disponível em: https://bitbucket.org/openid/ekyc-ida/wiki/Implementations.

MAINKA, C. et al. Sok: single sign-on security—an evaluation of openid connect. In: IEEE. **2017 IEEE European Symposium on Security and Privacy (EuroS&P)**. [S.l.], 2017. p. 251–266.

Ministério da Gestão e da Inovação em Serviços Públicos. 2021. Disponível em: http://faq-login-unico.servicos.gov.br/en/latest/_perguntasdafaq/oquee.html.

Ministério da Gestão e da Inovação em Serviços Públicos. 2021. Disponível em: https://www.gov.br/governodigital/pt-br/conta-gov-br/saiba-mais-sobre-os-niveis-da-conta-govbr/saiba-mais-sobre-os-niveis-da-conta-govbr.

NAVAS, J.; BELTRÁN, M. Understanding and mitigating openid connect threats. **Computers & Security**, v. 84, p. 1–16, 2019. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404818312781.

NIELSEN, H. et al. **Hypertext Transfer Protocol – HTTP/1.1**. RFC Editor, 1999. RFC 2616. (Request for Comments, 2616). Disponível em: https://www.rfc-editor.org/info/rfc2616.

OpenID Foundation. **eKYC & IDA Working Group - Overview**. 2022. Disponível em: https://openid.net/wg/ekyc-ida/.

OpenID Foundation. **OpenID Foundation - Certifications**. OpenID, 2023. Disponível em: https://openid.net/certification/.

Oracle. **JAR File Overview**. 2023. Disponível em: https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jarGuide.html.

PAIVA, D. P. D. Authentication modules for keycloak authentication server. 2018.

PARECKI, A.; WAITE, D. **OAuth 2.0 for Browser-Based Apps**. [S.l.], 2023. Work in Progress. Disponível em: https://datatracker.ietf.org/doc/draft-ietf-oauth-browser-based-apps/13/.

Postman, Inc. **What is Postman?** 2023. Disponível em: https://www.postman.com/product/what-is-postman/.

SAKIMURA, N.; BRADLEY, J.; JAY, E. **Financial-grade API Security Profile 1.0 - Part 1: Baseline**. OpenID, 2021. Disponível em: https://openid.net/specs/openid-financial-api-part-1-1_0-final.html.

SAKIMURA, N. et al. Openid connect core 1.0. **The OpenID Foundation**, p. S3, 2014.

SAKIMURA, N. et al. Openid connect discovery 1.0 incorporating errata set 1. **OpenID Foundation. Nov**, v. 8, 2014.

SAKIMURA, N. et al. Openid connect dynamic client registration 1.0. **OpenID Specification. OpenID Foundation**, 2014.

SCHARDONG, F.; CUSTÓDIO, R. Self-sovereign identity: A systematic review, mapping and taxonomy. **Sensors**, v. 22, n. 15, 2022. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/22/15/5641.

SCHARDONG, F. et al. Post-quantum electronic identity: Adapting openid connect and oauth 2.0 to the post-quantum era. In: BERESFORD, A. R.; PATRA, A.; BELLINI, E. (Ed.). **Cryptology and Network Security**. Cham: Springer International Publishing, 2022. p. 371–390. ISBN 978-3-031-20974-1.

SHARIF, A. et al. The eidas regulation: A survey of technological trends for european electronic identity schemes. **Applied Sciences**, v. 12, n. 24, 2022. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/12/24/12679.

SILVA, B. et al. Identificação eletrônica do registro civil do brasil. In: SBC. **Anais do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. [S.l.], 2023.

The Apache Software Foundation. **Welcome to Apache Maven**. 2023. Disponível em: https://maven.apache.org/.

THORGERSEN, S.; SILVA, P. I. **Keycloak-identity and access management for modern applications: harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications**. [S.l.]: Packt Publishing Ltd, 2021.

VARANASI, B. **Introducing Maven: A Build Tool for Today's Java Developers**. [S.l.]: Apress, 2019.

WRIGHT, A.; ANDREWS, H.; HUTTON, B. **JSON Schema Validation: A Vocabulary for Structural Validation of JSON**. 2022. Disponível em: https://json-schema.org/draft/2020-12/json-schema-validation.

WRIGHT, A. et al. **JSON Schema: A Media Type for Describing JSON Documents**. 2022. Disponível em: https://json-schema.org/draft/2020-12/json-schema-core.

APÊNDICE A – *CLAIMS* NO *OPENID CONNECT* 1.0

Abaixo, são listadas as possíveis *claims*, relacionadas aos dados do usuário, definidas na documentação do *OpenID Connect* 1.0, em conjunto com seus tipos e significados:

Tabela 1 – Lista de *claims* do usuário, como apresentado no *OpenID Connect* 1.0

Claim	**Tipo**	**Descrição**
`sub`	`string`	Identificador do usuário.
`name`	`string`	Nome completo.
`given_name`	`string`	Primeiro nome.
`family_name`	`string`	Sobrenome, ou último nome.
`middle_name`	`string`	Nome do meio.
`nickname`	`string`	Nome casual ou apelido.
`preferred_username`	`string`	Nome como prefere ser chamado pelo cliente.
`profile`	`string`	URL da página de perfil do usuário.
`picture`	`string`	URL da foto de perfil do usuário.
`website`	`string`	URL do *blog* ou página pessoal do usuário.
`email`	`string`	Endereço de *email* preferido do usuário.
`email_verified`	`boolean`	Verdadeiro, se o *email* foi verificado.
`gender`	`string`	Gênero.
`birthdate`	`string`	Data de nascimento.
`zoneinfo`	`string`	Fuso-horário.
`locale`	`string`	Idioma.
`phone_number`	`string`	Número de telefone.
`phone_number_verified`	`boolean`	Verdadeiro, se o número de telefone foi verificado.
`address`	`JSON`	Endereço, composto por:
`- formatted`	`string`	Endereço completo.
`- street_address`	`string`	Rua, número, código postal e outras informações.
`- locality`	`string`	Cidade ou localidade.
`- region`	`string`	Estado, província, prefeitura ou região.
`- postal_code`	`string`	Código postal.
`- country`	`string`	País.
`updated_at`	`number`	Data de última atualização do perfil.

Fonte: Sakimura et al. (2014a)

APÊNDICE B – NOVAS *CLAIMS* NO *OPENID CONNECT FOR IDENTITY ASSURANCE* 1.0

A seguir, são listadas novas *claims* relacionadas ao usuário, definidas pelo *OpenID Connect for Identity Assurance* 1.0, com seus tipos e significados:

Tabela 2 – Lista de *claims* do usuário, como apresentado no *OpenID Connect for Identity Assurance* 1.0

Claim	**Tipo**	**Descrição**
`address`	`JSON`	Novo campo adicionado:
`- country_code`	`string`	Código do país.
`place_of_birth`	`JSON`	Local de nascimento, contendo:
`- country`	`string`	País de nascimento.
`- locality`	`string`	Cidade ou localidade.
`- region`	`string`	Estado, província, prefeitura ou região.
`nationalities`	`array`	Lista de nacionalidades.
`birth_family_name`	`string`	Sobrenome, dado em seu nascimento.
`birth_given_name`	`string`	Nome, dado em seu nascimento.
`birth_middle_name`	`string`	Nome do meio, dado em seu nascimento
`salutation`	`string`	Saudação.
`title`	`string`	Título.
`msisdn`	`string`	Número de celular.
`also_known_as`	`string`	Nome artístico, nome religioso ou apelido/pseudônimo.

Fonte: Lodderstedt et al. (2022)

Implementação do protocolo *OpenID Connect for Identity Assurance 1.0* no *Keycloak*

Brendon Vicente Rocha Silva[1], **Ricardo F. Custódio**[1], **Frederico Schardong**[2]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)

[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)

Resumo. *Com o avanço das relações digitais, o aspecto da identificação digital segura torna-se crucial. Nesse sentido, protocolos como o OpenID Connect 1.0, que atua primariamente na identificação e autenticação de usuários em ambientes virtuais, se fazem muito necessários. O presente trabalho concretiza-se tendo em vista esforços recentes nesse campo, ou de maneira mais específica, a recente proposta do protocolo OpenID Connect for Identity Assurance 1.0 (OIDC4IDA). É descrito o esforço despendido para viabilizar sua compatibilidade com um dos principais provedores de identidade existentes: o Keycloak. A realização dessa implementação foi feita de maneira a manter a relevância e expandir o acesso do protocolo diante da rápida evolução tecnológica vigente.*

1. Introdução

Desde sua concepção nos anos 1960, a *internet* se tornou um pilar fundamental na sociedade moderna, revolucionando a comunicação, o trabalho e o entretenimento. Proporcionalmente, o número de informações sensíveis guardadas no meio virtual também tem aumentado cada vez mais, bem como a gravidade de potenciais fraudes e vazamentos. Os fatores supracitados implicam, por sua vez, na intensificação da discussão acerca do uso, proteção e armazenamento dessas informações. Um sinal explícito de tal fenômeno é a promulgação de leis e regulamentos voltadas especificamente para segurança de dados digitais, como o Regulamento Geral sobre a Proteção de Dados (*General Data Protection Regulation* (GDPR)), na União Européia, e a Lei Geral de Proteção de Dados Pessoais (LGPD), no Brasil [European Commission 2016, Brasil 2018]. Outro fator importante, no que diz respeito a esforços de segurança informacional no meio digital, é a criação de protocolos de segurança voltados à concessão de autorização digital e à manipulação de informações em identidades eletrônicas. Nesse cenário específico, podem ser citados alguns protocolos proeminentes, como o ***OAuth* 2.0** [Hardt 2012] e o ***OpenID Connect* 1.0** [Sakimura et al. 2014], que são vastamente utilizado em grandes sistemas e aplicações de renome ao redor do mundo [Mainka et al. 2017, Fett et al. 2016].

A extensão *OpenID Connect for Identity Assurance* 1.0 visa aprimorar ainda mais esses protocolos, trazendo funcionalidades voltadas para a verificação da autenticidade de informações do usuário. Esse trabalho foca no processo de implementação desse protocolo no ***Keycloak***, um dos principais provedores de identidade de código aberto, com o objetivo de facilitar sua adoção em casos de uso diversos. Para isso, foi determinado que a solução desenvolvida deveria ser modular, capaz de se adaptar a diferentes cenários e *open source*. Ao realizar essa implementação, espera-se fortalecer a infraestrutura de

segurança digital, contribuindo para um ambiente *online* mais seguro e confiável, e pavimentando o caminho para inovações futuras na gestão de identidades digitais e segurança de dados.

2. OpenID Connect for Identity Assurance 1.0

O *OpenID Connect for Identity Assurance* 1.0 é um protocolo que busca estender o *OpenID Connect* 1.0 [Lodderstedt and Fett 2019]. O propósito dessa especificação é permitir que dados a respeito do nível de garantia de informações do usuário sejam transmitidos, padronizados e suportados pelo *OpenID Connect* 1.0 [Sharif et al. 2022]. Essa funcionalidade é particularmente útil em aplicações que requerem alto grau de segurança, como serviços bancários e governamentais, por exemplo, tendo em vista regulamentações tais quais a LGPD e GPDR [Brasil 2018, European Commission 2016].

Um provedor de identidades, ao implementar o OIDC4IDA, deve ser capaz de fornecer informações relativas ao processo de validação dos dados de um usuário, como regulamentação sob a qual atua e sob a qual os dados foram validados, bem como evidências quanto à integridade dessas informações [Lodderstedt et al. 2022]. A especificação define uma série de regras, estruturas e mecanismos para que aplicações possam requisitar esses dados e determina minuciosamente como essas solicitações devem ser tratadas pelo provedor de identidades.

3. Trabalhos relacionados

Buscando fazer um mapeamento das soluções que comportassem o *OpenID Connect for Identity Assurance* 1.0, foi realizado um estudo de levantamento, com enfoque em bibliotecas, ferramentas e plataformas de gerenciamento de acesso e identidade, como o *Keycloak*, que alegassem comportar as funcionalidades descritas pela especificação. A própria equipe responsável pelo desenvolvimento do OIDC4IDA se propõe a realizar um mapeamento semelhante e mantém um repositório que lista os serviços que o incorporam [Lodderstedt et al. 2023]. Essa relação evidencia a escassez de trabalhos relacionados ao protocolo estudado, mencionando apenas três implementações de *software* e dois serviços em produção que afirmam conformidade com o protocolo, além de alguns sistemas considerando sua adoção.

Dentre os sistemas listados, o *Authlete* [Authlete 2023] é um serviço pago de identificação e autenticação de usuários baseado em nuvem. Ele simplifica a integração dos protocolos *OAuth* 2.0 e *OpenID Connect* 1.0 em aplicações, oferecendo APIs que atuam como servidores de autorização e eliminando a necessidade de um banco de dados dedicado. Além disso, ele disponibiliza gratuitamente uma biblioteca em Java que auxilia no desenvolvimento de soluções relacionadas à especificação. No entanto, por adotar um modelo de negócios pago, o *Authlete* distancia-se do escopo desse projeto.

Outra solução reconhecida oficialmente pela equipe do OIDC4IDA é o *Connect2id Server* [Connect2id 2023], que é gerenciado pela empresa búlgara *Connect2id* e busca facilitar o uso dos protocolos supracitados, de maneira similar ao *Authlete*. Uma biblioteca gratuita de integração em Java também é disponibilizada pela organização e conta com funcionalidades úteis para a adoção do protocolo. De maneira semelhante a solução anterior, o *Connect2id Server* foge ao escopo do projeto por ser um serviço em modalidade paga.

A última implementação indicada pela equipe de desenvolvimento do OIDC4IDA é o eKYC *Hub* [Identity First Tech 2021]. Esse projeto busca integrar o *Keycloak* com o protocolo *OpenID Connect for Identity Assurance* 1.0, diferenciando-se das soluções anteriores. Seu objetivo é utilizar a infraestrutura do *Keycloak* para processar requisições de atributos verificados de um servidor externo. No entanto, a documentação do projeto é limitada, e os links para informações adicionais e para o serviço de verificação de identidades *Passbase* estão desatualizados ou inoperantes. Este projeto, apesar de sua proposta modular, enfrenta desafios em termos de manutenção e confiabilidade, o que dificulta sua adoção em ambientes de produção. Além disso, por depender exclusivamente de um serviço externo, o eKYC *Hub* não configura-se como uma solução verdadeiramente modular e não atende aos requisitos do presente trabalho.

Por fim, uma busca por soluções alternativas relacionadas ao *Keycloak* e ao *OpenID Connect for Identity Assurance* 1.0 revelou uma proposta no repositório de desenvolvimento do *Keycloak* no *GitHub*. Esta solução, proposta em 2023, busca implementar funcionalidades nativas do protocolo no *Keycloak*, mas enfrenta problemas técnicos e de conformidade com a especificação. Além disso, ela propõe uma modificação no código-fonte do *Keycloak*, o que implica em desvantagens significativas, como a dependência de uma versão específica do sistema. Ademais, similar a solução anterior, essa implementação depende de um serviço externo para funcionar.

Tendo em vista a pesquisa realizada, portanto, conclui-se que nenhuma das soluções apresentadas cumpre, de forma satisfatória, com os requisitos definidos no escopo desse projeto: trazer suporte ao OIDC4IDA em uma ferramenta gratuita, de maneira modular e simples. Nesse aspecto, o presente trabalho justifica-se, pretendendo trazer avanços significativos em relação à adoção e popularização do protocolo estudado.

4. Desenvolvimento

Após análise e levantamento de recursos, foi possível dar inicio ao estágio de desenvolvimento de uma solução própria. A configuração inicial do ambiente de desenvolvimento envolveu a criação de uma instância do Keycloak usando contêineres *Docker* [Docker 2023], juntamente com a utilização da ferramenta `keycloak-config-cli` [Adorsys GmbH & Co. KG 2023] para configuração. Foi desenvolvido um *Service Provider* personalizado em um projeto *Maven* [The Apache Software Foundation 2023], e o código foi encapsulado em um arquivo *Java Archive* (JAR), que foi transferido para o ambiente do *Keycloak*. Ademais, um *shell script* foi criado para automatizar o processo de compilação e execução do projeto [Kernighan and Mashey 1979].

Após configurar o ambiente de desenvolvimento, deu-se início ao processo de implementação. A lógica de processamento da solução foi arquitetada para seguir as especificações do protocolo, incluindo funcionalidades como a validação de objetos JSON utilizando JSON *schemas*. Para o processamento de requisições, a biblioteca `authlete-java-common` foi utilizada, garantindo agilidade e maior conformidade da solução com as especificações do protocolo. Outra funcionalidade adicionada à solução, quando comparada aos outros trabalhos, foi permitir a escolha entre consumir os dados verificados diretamente do banco do *Keycloak* ou de um servidor de verificação externo, de acordo com as configurações do sistema.

A fase de testes da solução proposta foi conduzida utilizando a ferramenta *Post-*

man, que permite simular requisições HTTP e interagir com APIs [Postman, Inc. 2023]. Foram utilizados arquivos de exemplo disponíveis no repositório de desenvolvimento do *OpenID Connect for Identity Assurance*. Um ambiente de simulação também foi configurado no *Keycloak* através do `keycloak-config-cli`. Os testes abrangeram diversos cenários, incluindo exemplos de solicitações e respostas de atributos verificados, validação de documentos, requisições múltiplas e detalhes mínimos da especificação. Essa bateria de testes validou a solução proposta, identificou possíveis *bugs* e permitiu realizar correções necessárias para assegurar o bom funcionamento da implementação.

5. Conclusões e trabalhos futuros

Durante o desenvolvimento deste trabalho, foi realizado um amplo levantamento de implementações do protocolo *OpenID Connect for Identity Assurance 1.0*, permitindo a análise de suas características, vantagens e limitações. Isso proporcionou um melhor entendimento das especificações técnicas do protocolo, bem como dos desafios relacionados à sua adoção e popularização.

A pesquisa identificou uma lacuna no mercado em relação à implementação do protocolo no *Keycloak*, um dos sistemas de gerenciamento de identidade eletrônica mais reconhecidos. Com base nessa identificação, a proposta de desenvolver uma solução para integrar o protocolo ao *Keycloak* foi concebida. A escolha de desenvolver uma solução integrada ao*Keycloak* é significativa, pois torna o suporte ao OIDC4IDA acessível em uma ferramenta amplamente reconhecida, *open-source* e gratuita.

A solução não apenas atendeu aos requisitos iniciais, mas também ofereceu funcionalidades adicionais. A abordagem modular da solução traz várias vantagens, como a facilidade de integração com sistemas em produção e a interoperabilidade entre diferentes versões do *Keycloak*. Além disso, permite a modificação do código da solução, sem afetar os arquivos do *Keycloak*, possibilitando adaptações ágeis e simplificadas para atender a casos de uso específicos. Um destaque importante da solução desenvolvida é a utilização da biblioteca `authlete-java-common`, que garante maior conformidade com o protocolo e é recomendada pela sua equipe de desenvolvimento. Ademais, a validação dos artefatos por meio de JSON *schemas* também contribui para a robustez da implementação.

Por fim, a capacidade de escolher entre consumir atributos verificados diretamente do banco de dados do *Keycloak* ou de um servidor externo de verificação é uma funcionalidade única da implementação, que não foi encontrada em outras soluções mapeadas. Isso contribui para a flexibilidade da solução, eliminando a necessidade estrita de um servidor externo de verificação.

Dessa forma, a implementação proposta contribui para a adoção e popularização do OIDC4IDA. Como desdobramentos futuros, planeja-se aprimorar a solução com testes automatizados e considera-se implantá-la em um sistema de identificação eletrônica de alcance nacional, a Autenticação Eletrônica do Registro Civil do Brasil (IdRC), para promover o uso da solução desenvolvida e, subsequentemente, do protocolo.

Referências

Adorsys GmbH & Co. KG (2023). keycloak-config-cli.

Authlete (2023). Authlete - homepage.

Brasil (2018). Lei nº 13.709, de 14 de agosto de 2018. *Diário Oficial [da] República Federativa do Brasil*.

Connect2id (2023). Connect2id server.

Docker (2023). Docker - homepage.

European Commission (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance).

Fett, D., Küsters, R., and Schmitz, G. (2016). A comprehensive formal security analysis of oauth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1204–1215.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749.

Identity First Tech (2021). ekyc hub.

Kernighan, B. W. and Mashey, J. R. (1979). The unix™ programming environment. *Software: Practice and Experience*, 9(1):1–15.

Lodderstedt, T. and Fett, D. (2019). Openid connect for identity assurance.

Lodderstedt, T., Fett, D., Haine, M., Pulido, A., Lehmann, K., and Koiwai, K. (2022). Openid connect for identity assurance 1.0.

Lodderstedt, T., Sanz, M., and Haine, M. (2023). Openid connect for identity assurance implementations.

Mainka, C., Mladenov, V., Schwenk, J., and Wich, T. (2017). Sok: single sign-on security—an evaluation of openid connect. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 251–266. IEEE.

Postman, Inc. (2023). What is postman?

Sakimura, N., Bradley, J., Jones, M., De Medeiros, B., and Mortimore, C. (2014). Openid connect core 1.0. *The OpenID Foundation*, page S3.

Sharif, A., Ranzi, M., Carbone, R., Sciarretta, G., Marino, F. A., and Ranise, S. (2022). The eidas regulation: A survey of technological trends for european electronic identity schemes. *Applied Sciences*, 12(24).

The Apache Software Foundation (2023). Welcome to apache maven.