



Universidade do Minho
Escola de Engenharia

Tiago Passos Rodrigues

**Um sistema para registo seguro do
consentimento de processamento
de dados pessoais**



Universidade do Minho
Escola de Engenharia

Tiago Passos Rodrigues

**Um sistema para registo seguro do
consentimento de processamento
de dados pessoais**

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação de
João Marco Cardoso da Silva
Ana Luísa Parreira Nunes Alonso

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho:

[Caso o autor pretenda usar uma das licenças Creative Commons, deve escolher e deixar apenas um dos seguintes ícones e respetivo lettering e URL, eliminando o texto em itálico que se lhe segue. Contudo, é possível optar por outro tipo de licença, devendo, nesse caso, ser incluída a informação necessária adaptando devidamente esta minuta]



CC BY

<https://creativecommons.org/licenses/by/4.0/> *[Esta licença permite que outros distribuam, remixem, adaptem e criem a partir do seu trabalho, mesmo para fins comerciais, desde que lhe atribuam o devido crédito pela criação original. É a licença mais flexível de todas as licenças disponíveis. É recomendada para maximizar a disseminação e uso dos materiais licenciados.]*



CC BY-SA

<https://creativecommons.org/licenses/by-sa/4.0/> [Esta licença permite que outros remisturem, adaptem e criem a partir do seu trabalho, mesmo para fins comerciais, desde que lhe atribuam o devido crédito e que licenciem as novas criações ao abrigo de termos idênticos. Esta licença costuma ser comparada com as licenças de software livre e de código aberto «copyleft». Todos os trabalhos novos baseados no seu terão a mesma licença, portanto quaisquer trabalhos derivados também permitirão o uso comercial. Esta é a licença usada pela Wikipédia e é recomendada para materiais que seriam beneficiados com a incorporação de conteúdos da Wikipédia e de outros projetos com licenciamento semelhante.]



CC BY-ND

<https://creativecommons.org/licenses/by-nd/4.0/> [Esta licença permite que outras pessoas usem o seu trabalho para qualquer fim, incluindo para fins comerciais. Contudo, o trabalho, na forma adaptada, não poderá ser partilhado com outras pessoas e têm que lhe ser atribuídos os devidos créditos.]



CC BY-NC

<https://creativecommons.org/licenses/by-nc/4.0/> [Esta licença permite que outros remisturem, adaptem e criem a partir do seu trabalho para fins não comerciais, e embora os novos trabalhos tenham de lhe atribuir o devido crédito e não possam ser usados para fins comerciais, eles não têm de licenciar esses trabalhos derivados ao abrigo dos mesmos termos.]



CC BY-NC-SA

<https://creativecommons.org/licenses/by-nc-sa/4.0/> [Esta licença permite que outros remisturem, adaptem e criem a partir do seu trabalho para fins não comerciais, desde que lhe atribuam a si o devido crédito e que licenciem as novas criações ao abrigo de termos idênticos.]



CC BY-NC-ND

<https://creativecommons.org/licenses/by-nc-nd/4.0/> [Esta é a mais restritiva das nossas seis licenças principais, só permitindo que outros façam download dos seus trabalhos e os com-

partilhem desde que lhe sejam atribuídos a si os devidos créditos, mas sem que possam alterá-los de nenhuma forma ou utilizá-los para fins comerciais.]

Agradecimentos

Escreva aqui os seus agradecimentos. Não se esqueça de mencionar, caso seja esse o caso, os projetos e bolsas dos quais se beneficiou enquanto fazia a sua investigação. Pergunte ao seu orientador sobre o formato específico a ser usado. (As agências de financiamento são bastante rigorosas quanto a isso.)

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, Braga, outubro 2025

Tiago Passos Rodrigues

Resumo

A gestão de consentimento de dados é uma questão central no contexto de regulamentações como o **Regulamento Geral de Proteção de Dados (RGPD)**, que visa garantir a privacidade dos utilizadores. Esta dissertação explora as **Plataformas de Gestão de Consentimento (CMP)** existentes, destacando as suas características, limitações e potencialidades para a criação de soluções mais auditáveis e transparentes. O objetivo é desenvolver uma solução que permita não apenas recolher consentimentos de forma eficaz, mas também auditar as escolhas dos utilizadores. Inicialmente, foi realizada uma análise do trabalho relacionado, que incluiu não apenas plataformas como Osano, Cookiebot, Klaro.js e outros, além de soluções avançadas como o Google Consent Mode, mas também abordagens complementares para a gestão de consentimento, nomeadamente baseadas em tecnologias como *blockchain* e contratos inteligentes. Verificou-se que as **CMPs** existentes, embora robustas, apresentam limitações no que toca à auditoria e à transparência do fluxo de dados entre clientes e servidores. Neste projeto é demonstrado e feita uma prova de conceito de um sistema que integre a recolha de consentimento com mecanismos de auditoria. A abordagem permite garantir que os consentimentos recolhidos são armazenados de forma verificável, como também imutável, assegurando conformidade com as normas aplicáveis. Este estudo visa contribuir para o desenvolvimento de uma solução que reforça a confiança dos utilizadores no tratamento dos seus dados, promovendo uma maior transparência e conformidade regulatória.

Palavras-chave **CMP**, **RGPD**, auditoria, transparência, auditabilidade, imutabilidade, privacidade, *open source*

Abstract

Data consent management is a central issue in the context of regulations such as the **General Data Protection Regulation (GDPR)**, which aim to ensure user privacy. This dissertation explored existing consent management platforms (CMPs), highlighting their features, limitations, and potential for creating more auditable and transparent solutions. The objective was to develop a solution that not only effectively collected consents but also audited user choices. Initially, a related work analysis was carried out, which included not only platforms such as Osano, Cookiebot, Klaro.js, and others, as well as advanced solutions like Google Consent Mode, but also complementary approaches to consent management, namely those based on technologies such as *blockchain* and smart contracts. It was found that existing **CMPs**, although robust, presented limitations regarding the auditability and transparency of data flows between clients and servers. In this project, a proof of concept was implemented, integrating consent collection with auditing mechanisms. The approach ensured that collected consents were stored in a verifiable manner, as well being immutable, guaranteeing compliance with applicable regulatory standards. This study contributed to the development of a solution that reinforced user trust in data processing, promoting greater transparency and regulatory compliance.

Keywords **CMP**, **GDPR**, auditing, transparency, auditability, immutability, privacy, *open source*

Conteúdo

1	Introdução	1
1.1	Motivação	1
1.2	Objetivos	2
1.3	Contribuições	2
1.4	Estrutura da dissertação	3
2	Trabalho Relacionado	4
2.1	Plataformas de Gestão de Consentimento (CMP)	4
2.1.1	Fluxo de Implementação de uma CMP	5
2.2	Limitações das CMPs Existentes	9
2.2.1	Discussão	10
2.3	Regulamento Geral de Proteção de Dados (RGPD)	11
2.4	<i>Blockchain</i> e Contratos Inteligentes	12
3	Visão Geral da Architectura	16
3.1	Componentes	17
3.2	Interacção Entre Componentes	17
3.3	Lógica de Funcionamento	19
3.4	Modelo de confiança	19
3.4.1	Certificados digitais	19
3.4.2	Assinatura Digital	20
3.5	Registo do consentimento	20
3.6	Benefícios da Architectura	21
4	Implementação da solução	22
4.1	Open-Source	22

4.1.1	A Necessidade de CMPs Open-Source	23
4.2	Servidor e Website	23
4.3	JWS	25
4.3.1	Descrição Geral (Baseado no RFC 7515)	25
4.3.2	Aplicação ao Caso de Consentimento	26
4.3.3	Vantagens neste contexto	26
4.4	Extensão no Cliente	27
4.5	Gestão de Certificados	28
4.6	Fluxo de Consentimento	29
4.7	Registo do consentimento	31
5	Conclusões e trabalho futuro	34
5.1	Conclusões	34
5.2	Trabalho futuro	35
5.2.1	Complementação do fluxo com <i>blockchain</i> e contratos inteligentes	35
5.2.2	Integração com keychain do sistema do cliente	36
5.2.3	Disponibilização da solução como projeto Open-source	36
6	Avaliação	37
6.1	Avaliação Funcional	37
6.2	Avaliação de Desempenho	40
6.3	Discussão dos Resultados	40
I	Apêndices	44
A	Trabalho de apoio	45
B	Detalhes dos resultados	47
C	Listings	48
D	Ferramentas	49

Lista de Figuras

1	Fluxo de implementação de uma CMP num site.	6
2	Exemplo de visualização do consentimento na <i>dashboard</i> do CookieChimp.	8
3	Exemplo de banner de consentimento exibido ao utilizador.	9
4	Diagrama do protocolo	18
5	Banner padrão do Klaro.js	30
6	Diagrama do protocolo	31
7	Extensão carregada no browser	38

Lista de Tabelas

1	Comparação das principais características das CMPs existentes	10
2	Resultados de desempenho em diferentes cenários de payload.	40

Capítulo 1

Introdução

Com a digitalização em crescimento e a utilização crescente de plataformas online, a quantidade de dados pessoais recolhidos dos utilizadores tem aumentado exponencialmente. Contudo, muitos utilizadores desconhecem como esses dados são recolhidos, processados e partilhados pelos servidores. Quando um utilizador acede a um site, ele frequentemente não tem uma visão clara sobre o que acontece com os dados fornecidos nem se o servidor recebeu exatamente aquilo a que deu consentimento. Este cenário levanta questões de privacidade, especialmente no que diz respeito ao cumprimento de regulamentações como o **Regulamento Geral de Proteção de Dados (RGPD)** e outras legislações de privacidade.

No contexto atual, as Plataformas de Gestão de Consentimento (**CMPs**) surgem como ferramentas que dizem assegurar às plataformas das empresas que assim podem cumprir o **RGPD** (Santos et al., 2021), permitindo a recolha e gestão do consentimento dos utilizadores para a utilização dos seus dados. Contudo, as **CMPs** tradicionais, muitas das quais são soluções fechadas, apresentam limitações em termos de transparência e auditoria. Estas plataformas não fornecem uma visão clara sobre o que realmente acontece com os dados depois de o utilizador ter dado o seu consentimento, nem se a informação transmitida ao servidor está em conformidade com a escolha do utilizador. A falta de uma auditoria transparente torna difícil garantir que as empresas estão a respeitar o consentimento dos utilizadores e a cumprir as regulamentações de privacidade.

1.1 Motivação

A motivação para este projeto surge precisamente da necessidade de permitir a auditoria desse fluxo de dados, oferecendo uma forma de garantir que as escolhas do utilizador são efetivamente respeitadas. Uma das grandes limitações das **CMPs** tradicionais é o facto de muitas delas serem soluções fechadas, dificultando a compreensão e auditoria do que acontece com os dados. Por isso, optou-se por explorar soluções de **CMP open source**, que, ao serem mais acessíveis, permitem uma maior transparência e

auditabilidade dos dados recolhidos. Através dessas plataformas, é possível compreender melhor os fluxos de dados e garantir que o processo de consentimento seja claro e auditável.

1.2 Objetivos

O objetivo deste trabalho é criar uma solução que permita não só recolher e gerir o consentimento dos utilizadores, mas também auditar o processo de forma transparente e verificável. Uma possível solução para isso é o uso de tecnologias como assinaturas digitais, certificados, que, ao permitir registar de forma imutável as interações e escolhas dos utilizadores, pode assegurar a conformidade e aumentar a confiança no processo. O principal objetivo deste trabalho é, portanto, desenhar e prototipar um sistema que ajude na gestão do consentimento do utilizador uma vez que se integra a recolha de consentimento com mecanismos de auditoria transparentes, utilizando estratégias criptográficas, de forma a criar transparência entre as decisões dos utilizadores e os dados que são armazenados nas **CMP** e que esteja conforme o **RGPD**.

Para atingir este objetivo, será realizada inicialmente uma revisão de literatura sobre **CMPs**, **RGPD** e a utilização de *blockchain* para auditoria de consentimento e integração de contratos inteligentes. Em seguida, proceder-se-á à análise das abordagens existentes para plataformas de gestão de consentimento, identificando as suas limitações e as melhores alternativas para o nosso sistema. Com base nesta análise, foi desenhado um modelo arquitetural para uma plataforma que integre auditoria transparente e registo imutável de consentimentos. Posteriormente, foi desenvolvido uma prova de conceito utilizando essa mesma arquitetura, com a integração um **CMP** Open-source. O sistema foi testado e otimizado, avaliando-se o seu desempenho, segurança e usabilidade.

1.3 Contribuições

As contribuições desta dissertação podem ser agrupadas em dois níveis complementares: conceptual e prático.

No plano conceptual, destaca-se o desenho de uma arquitetura genérica para a gestão de consentimento. Esta arquitetura foi concebida de forma modular e independente de tecnologias específicas, permitindo que qualquer pessoa ou organização a possa implementar segundo os seus próprios requisitos. Para reforçar a generalidade da proposta, optou-se por separar a descrição da arquitetura da implementação concreta, salientando a sua reusabilidade e extensibilidade.

No plano prático, apresenta-se a proposta e a implementação de uma solução funcional que mate-

realiza a arquitetura definida. O protótipo desenvolvido baseia-se em tecnologias *open-source* e integra mecanismos de assinatura digital, certificados e *JSON Web Signatures* (JWS), demonstrando a viabilidade da abordagem proposta. Entre os contributos técnicos concretos incluem-se a criação de uma extensão de navegador, o desenvolvimento de um servidor, a integração com o **CMP** *open-source* Klaro.js e a gestão de certificados, garantindo que o processo de consentimento é registado de forma transparente e auditável.

Em conjunto, estas contribuições oferecem não só uma abordagem para a recolha e auditoria de consentimentos, mas também uma solução prática que pode ser replicada e adaptada em diferentes contextos.

1.4 Estrutura da dissertação

A presente dissertação encontra-se organizada em três partes principais, compreendendo um total de seis capítulos.

Na primeira parte é introduzido o problema (Capítulo 1), bem como a motivação, os objectivos e o estudo do trabalho relacionado, apresentando algumas das soluções existentes para problemas semelhantes (Capítulo 2).

A segunda parte descreve os contributos principais do trabalho, incluindo o desenho detalhado da arquitectura proposta (Capítulo 3), a implementação da solução e a respectiva análise (Capítulo 4), e as conclusões gerais acompanhadas de perspectivas de trabalho futuro (Capítulo 5).

A terceira parte é dedicada à avaliação, onde se discute o funcionamento e a validação da solução proposta (Capítulo 6).

Por fim, os apêndices apresentam material de apoio, detalhes complementares dos resultados experimentais, listagens de código relevante e documentação das ferramentas utilizadas.

Capítulo 2

Trabalho Relacionado

A gestão de consentimento é um dos principais desafios no tratamento de dados pessoais, especialmente face ao crescimento exponencial da digitalização e às regulamentações como o **RGPD**. As **CMPs** surgem como uma solução prática para facilitar a recolha e a gestão do consentimento do utilizador, assegurando a conformidade com estas legislações. No entanto, enquanto as **CMPs** oferecem mecanismos para gerir o consentimento, existem limitações em termos de transparência e auditabilidade.

2.1 Plataformas de Gestão de Consentimento (CMP)

As **Plataformas de Gestão de Consentimento (CMP)** desempenham um papel essencial ao fornecer aos *titulares dos dados* uma forma clara e explícita de consentir com o tratamento dos seus dados pessoais. Estas plataformas surgiram como resposta à crescente necessidade de garantir que os direitos dos Titulares dos Dados sejam respeitados no contexto da recolha e tratamento dos seus dados, especialmente face às exigências do **RGPD**. A principal função das **CMPs** é proporcionar uma solução eficaz e transparente para a obtenção, armazenamento e gestão do consentimento dos Titulares dos Dados, garantindo que as organizações cumpram as obrigações legais impostas pelas regulamentações de privacidade. Estas plataformas permitem que os Titulares dos Dados tenham controlo sobre o uso dos seus dados pessoais e são devidamente informados sobre as práticas de tratamento destes mesmos.

Entre as soluções mais populares encontrou-se plataformas como **Osano**, **Cookiebot**, **Tarteaucitron.js**, **Klaro.js** e **Consent Manager**. Cada uma destas plataformas oferece funcionalidades específicas, mas todas têm em comum o foco na conformidade com as regulamentações de privacidade, como o **RGPD**.

- **Osano**: Oferece uma interface intuitiva de personalização e um sistema eficiente de gestão de preferências [Osano \(2025\)](#). Inclui recursos avançados como armazenamento local de preferências e bloqueio automático de rastreamento não autorizado. Contudo, por ser uma solução proprietária, apresenta limitações na verificação do processamento de dados entre o navegador e o servidor.

- **Cookiebot:** Fornece uma solução robusta que analisa automaticamente até 10.000 páginas por domínio [A/S \(2025\)](#). Implementa um sistema inteligente para detetar mudanças nos *cookies* e rastreadores do site. A principal limitação está na sua natureza fechada do código, que dificulta verificações independentes do funcionamento interno.
- **Tarteacitron.js & Klaro.js:** São soluções de código aberto que se destacam pela flexibilidade [Tarteacitron \(2025\)](#). O Tarteacitron.js permite extensões personalizadas e um carregamento otimizado de *scripts*, enquanto o Klaro.js oferece um sistema flexível de armazenamento de preferências e suporte a múltiplos idiomas. Ambos permitem personalização completa do controlo de *cookies*.
- **Consent Manager:** Utiliza uma estrutura moderna que permite sincronização instantânea das preferências do utilizador entre diferentes *tabs* do navegador [Manager \(2025\)](#). Embora ofereça recursos avançados de gestão, devido à sua natureza fechada, limita a possibilidade de auditorias independentes.
- **CookieChimp:** Diferencia-se pela facilidade de integração com outros sistemas e pela gestão eficiente de consentimento [CookieChimp \(2025\)](#). Disponibiliza uma **API** RESTful bem documentada que permite integração simples com sistemas externos.

Apesar dessas soluções estarem bem posicionadas para garantir a conformidade legal, um problema recorrente nas **CMP**s existentes é a falta de visibilidade sobre os dados transmitidos. Os utilizadores podem não saber exatamente o que acontece com as suas informações depois de fornecerem o consentimento. Não há uma forma clara e acessível para verificar se as escolhas feitas são efetivamente respeitadas, o que levanta questões sobre a transparência e a auditoria.

2.1.1 Fluxo de Implementação de uma **CMP**

O fluxo típico para garantir essa conformidade pode ser representado no diagrama de atividades da Figura 1. Este diagrama ilustra de forma genérica a integração de uma **CMP** num website. No âmbito da pesquisa realizada sobre soluções existentes, este fluxo foi exemplificado com a plataforma CookieChimp, utilizada apenas para efeitos de análise comparativa.

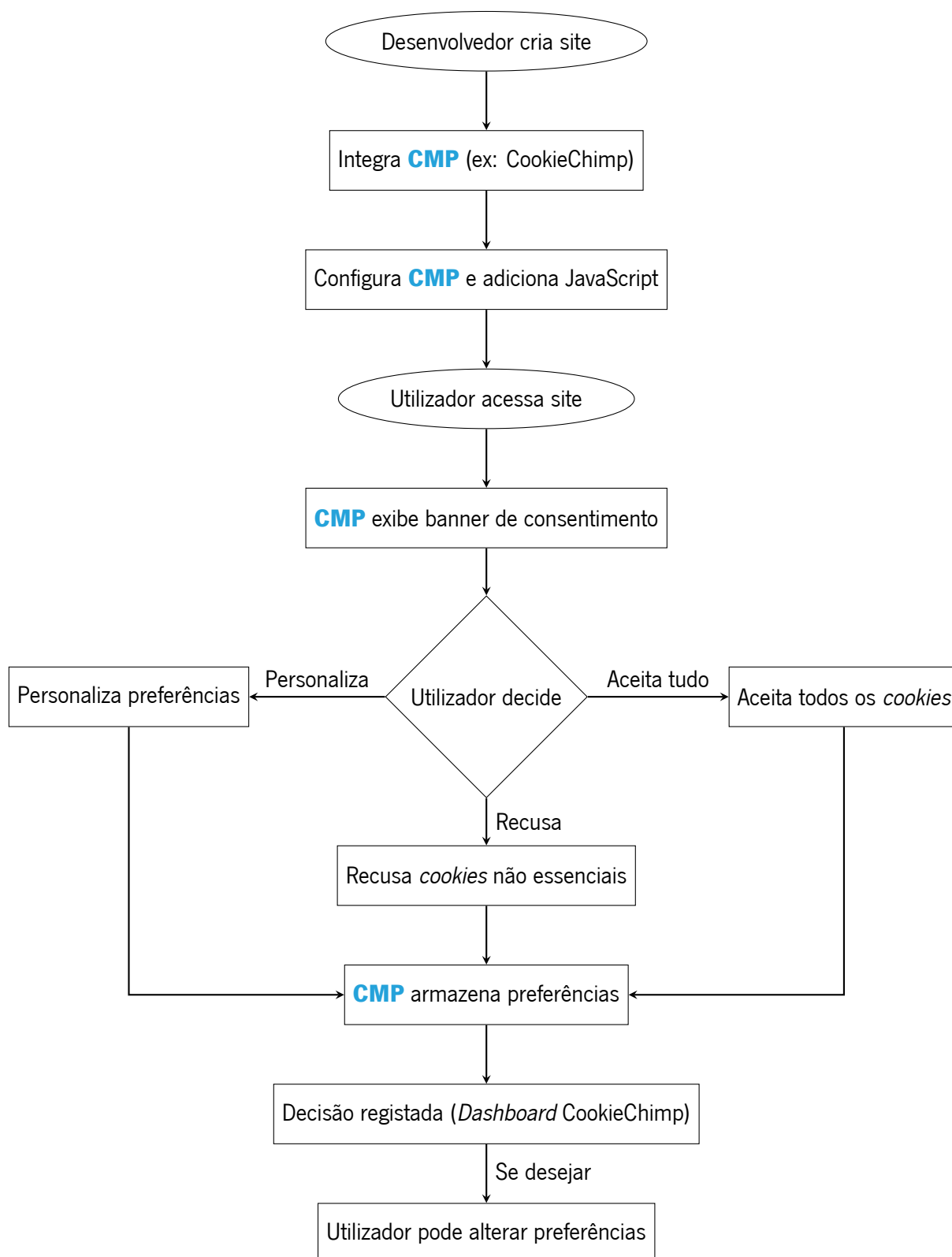


Figura 1: Fluxo de implementação de uma **CMP** num site.

O processo pode ser descrito da seguinte forma:

1. O desenvolvedor cria um site.
2. Para cumprir o **RGPD**, decide integrar uma **CMP**, como o **CookieChimp**.
3. Escolhe uma plataforma e configura a ferramenta escolhida, adicionando o seu código JavaScript ao site.
4. Quando um utilizador acessa o site, o *plugin* da **CMP** exibe um *banner* (fig. 3) solicitando consentimento.
5. O utilizador pode:
 - Aceitar todos os *cookies* e tecnologias de rastreamento.
 - Recusar todos os *cookies* não essenciais.
 - Personalizar as suas preferências, seleccionando categorias específicas (ex: marketing, estatísticas, essenciais).
6. A **CMP** armazena as preferências do utilizador e bloqueia os *cookies* de terceiros até que o consentimento seja dado.
7. A decisão do utilizador é registada e pode ser consultado pelo criador do site, via dashboard do **CookieChimp** como pode ser visto na figura 2.


Consent Information		Visitor Actions Granted Consents																													
Consent ID	4f4d888f-b7f0-11ef-80d9-f175c05cbc30	Visitor's accepted services & cookies.																													
IP Address	2001:818:e739:b800:fc22:a b07:ca35:XXXX	CookieChimp																													
IP Country	 Portugal	<table> <tr> <th>Name</th><th>Company</th><th>Description</th><th>Duration</th></tr> <tr> <td>cookiechimp_pref_*</td><td>CookieChimp</td><td>Used to store the user's cookie consen...</td><td>6 months</td></tr> <tr> <td>cookiechimp_uuid</td><td>CookieChimp</td><td>Used to identify unique site visitors.</td><td>Session</td></tr> </table>		Name	Company	Description	Duration	cookiechimp_pref_*	CookieChimp	Used to store the user's cookie consen...	6 months	cookiechimp_uuid	CookieChimp	Used to identify unique site visitors.	Session																
Name	Company	Description	Duration																												
cookiechimp_pref_*	CookieChimp	Used to store the user's cookie consen...	6 months																												
cookiechimp_uuid	CookieChimp	Used to identify unique site visitors.	Session																												
IP Region	Porto	WordPress																													
User ID	---	<table> <tr> <th>Name</th><th>Company</th><th>Description</th><th>Duration</th></tr> <tr> <td>comment_author_*</td><td>WordPress</td><td>Used to store the user's name and emai...</td><td>1 year</td></tr> <tr> <td>WordPress-*</td><td>WordPress</td><td>Used to store authentication details.</td><td>Session</td></tr> <tr> <td>wordpress_logged_in_*</td><td>WordPress</td><td>Indicates when you're logged in, and w...</td><td>Session</td></tr> <tr> <td>wordpress_sec_*</td><td>WordPress</td><td>Used to store your authentication details.</td><td>Session</td></tr> <tr> <td>wordpress_test_cookie</td><td>WordPress</td><td>Used to check if cookies are enabled o...</td><td>Session</td></tr> <tr> <td>wp-settings-*</td><td>WordPress</td><td>Used to persist a user's wp-admin conf...</td><td>1 year</td></tr> </table>		Name	Company	Description	Duration	comment_author_*	WordPress	Used to store the user's name and emai...	1 year	WordPress-*	WordPress	Used to store authentication details.	Session	wordpress_logged_in_*	WordPress	Indicates when you're logged in, and w...	Session	wordpress_sec_*	WordPress	Used to store your authentication details.	Session	wordpress_test_cookie	WordPress	Used to check if cookies are enabled o...	Session	wp-settings-*	WordPress	Used to persist a user's wp-admin conf...	1 year
Name	Company	Description	Duration																												
comment_author_*	WordPress	Used to store the user's name and emai...	1 year																												
WordPress-*	WordPress	Used to store authentication details.	Session																												
wordpress_logged_in_*	WordPress	Indicates when you're logged in, and w...	Session																												
wordpress_sec_*	WordPress	Used to store your authentication details.	Session																												
wordpress_test_cookie	WordPress	Used to check if cookies are enabled o...	Session																												
wp-settings-*	WordPress	Used to persist a user's wp-admin conf...	1 year																												
Accept Type	<input checked="" type="radio"/> All																														
GPC Preference	---																														
Accepted Categories	essential marketing																														
Rejected Categories	---																														
Accepted At	2025-01-20 21:48:23 UTC																														

Figura 2: Exemplo de visualização do consentimento na *dashboard* do CookieChimp.

- Se o utilizador desejar alterar as preferências, pode fazê-lo através de um botão de configuração acessível no site.

Desta forma, as **CMPs** garantem transparência e conformidade com as regulamentações de privacidade, permitindo que os utilizadores tenham maior controlo sobre os seus dados pessoais.

Cookie usage ✕

We use cookies to ensure the basic functionalities of the website and to enhance your online experience. You can choose for each category to opt-in/out whenever you want. For more details relative to cookies and other sensitive data, please read the full **privacy policy**.

^ **Essential** Always Enabled ☒

Essential cookies are necessary for features that are essential to your use of our site or services, such as account login, authentication and site security.

▼ **CookieChimp** Always Enabled ☒

▼ **WordPress** Always Enabled ☒

^ **Marketing** ☒

Marketing cookies make it possible to show you more relevant social media content and advertisements on our website and other platforms.

Save Preferences Reject Accept All

Figura 3: Exemplo de banner de consentimento exibido ao utilizador.

2.2 Limitações das **CMPs** Existentes

As **CMPs** desempenham um papel crucial ao fornecer aos utilizadores uma forma de consentir explicitamente com o uso dos seus dados pessoais. Entre as soluções mais populares, destacam-se **Osano**, **Cookiebot**, **Tarteaucitron.js**, **Klaro.js**, **Consent Manager** e **CookieChimp**. Cada uma destas plataformas oferece funcionalidades específicas e foca-se na conformidade com regulamentações de privacidade, como o **RGPD**. No entanto, apesar da sua importância, essas plataformas apresentam várias limitações que comprometem a transparência, a auditabilidade e a confiança no processamento de dados.

Para uma gestão de consentimento mais eficiente e alinhada com as expectativas dos utilizadores e as exigências regulamentares, certas características são fundamentais. Por exemplo, a abertura do código-fonte (*open source*) é uma característica valorizada, pois permite uma auditoria mais rigorosa do

processo, garantindo que os utilizadores possam verificar a integridade da plataforma. Além disso, a *auditabilidade* é uma qualidade essencial, uma vez que possibilita confirmar que as escolhas dos utilizadores são efetivamente respeitadas, fornecendo uma camada adicional de confiança ao assegurar que ele tem clareza sobre como as suas informações são tratadas.

A *intuitividade* da interface também é importante para garantir que os utilizadores consigam facilmente compreender e controlar as suas preferências de consentimento. Complementarmente, a *personalização* das configurações de consentimento oferece maior flexibilidade, permitindo que as plataformas possam ser adaptadas a diferentes contextos organizacionais, respeitando as necessidades específicas de cada caso. Por fim, a presença de uma *API RESTful* permite que a **CMP** se integre facilmente com outros sistemas, o que é particularmente relevante em ambientes corporativos mais complexos.

A tabela a seguir (1) compara algumas das plataformas mais populares, destacando as suas capacidades em relação a estas características-chave e as suas limitações:

Tabela 1: Comparação das principais características das **CMPs** existentes

Característica	Osano	Cookiebot	Tarteaucitron.js	Klaro.js	Consent Manager	CookieChimp
<i>open-source</i>			X	X		
Auditabilidade						
Intuitivo	X	X	X		X	X
Personalização		X	X	X		X
RESTful API		X			X	X

2.2.1 Discussão

Além das limitações específicas de cada plataforma, existem problemas mais gerais que afetam as **CMPs** tradicionais:

Primeiramente, verifica-se uma **falta de transparência** em como os dados dos utilizadores são processados e armazenados após o consentimento. Muitas plataformas não oferecem uma visão clara aos utilizadores, que frequentemente desconhecem se o consentimento foi transmitido corretamente ao servidor ou se as suas escolhas estão a ser respeitadas.

Outra limitação crítica é a **ausência de mecanismos robustos de auditoria**. As **CMPs** existentes não permitem uma verificação em tempo real do cumprimento das escolhas do utilizador. Esta ausência torna difícil garantir que os dados recolhidos são processados de forma consistente com os regulamentos de privacidade, como o **RGPD**.

Por fim, destaca-se o facto de que muitas soluções disponíveis no mercado são **soluções fechadas**. Isto significa que o código-fonte não está acessível ao público, impossibilitando auditorias independentes ou personalizações técnicas por terceiros. Este aspeto não apenas limita a flexibilidade técnica, mas também reduz a confiança geral nos processos internos das plataformas.

Embora estas **CMP**s cumpram aspetos básicos de conformidade regulatória, tornam-se insuficientes para os desafios modernos de gestão de consentimento. Assim, a criação de uma solução mais robusta, aberta e transparente surge como uma necessidade premente, capaz de superar essas limitações e estabelecer novos padrões de confiança no tratamento de dados pessoais.

2.3 Regulamento Geral de Proteção de Dados (RGPD)

O **RGPD** é uma legislação da **União Europeia (UE)** que visa garantir uma maior proteção e privacidade dos dados pessoais dos cidadãos da **UE**.

Além de estabelecer normas rigorosas para o tratamento de dados, o **RGPD** reforça os direitos dos titulares dos dados, incluindo o direito de acesso, retificação, remoção (*direito a ser esquecido*), portabilidade, limitação do tratamento e oposição. As organizações são também obrigadas a obter o consentimento explícito e informado dos utilizadores para o processamento dos seus dados pessoais. (Daudén-Esmel et al., 2024)

O **RGPD** identifica quatro intervenientes principais no seu quadro legal (European Parliament and Council, 2016):

- Titular dos Dados (**Data Subject (DS)**): A pessoa natural identificada ou identificável de quem as entidades podem recolher informações pessoais.
- Responsável pelo Tratamento (**Data Controller (DC)**): A pessoa natural ou entidade legal que determina as finalidades e os meios pelos quais os dados pessoais são processados.
- Destinatário dos Dados (**Data Recipient (DR)**): A pessoa natural ou entidade legal para a qual os dados pessoais são divulgados (pode ser o próprio **DC** ou um terceiro).
- Subcontratante (**Data Processor (DP)**): A pessoa natural ou entidade legal que processa dados pessoais em nome do **DC**.

O principal objetivo do **RGPD** é garantir direitos de privacidade específicos aos titulares dos dados, assegurando que os seus dados pessoais “só podem ser recolhidos legalmente, sob condições estritas

e para um propósito legítimo”. O regulamento procura devolver o controlo total sobre os dados aos seus titulares. Adicionalmente, o **RGPD** trouxe benefícios relevantes, como o aumento da consciência sobre a segurança e proteção de dados e a capacitação dos consumidores para controlar as suas preferências e participar ativamente na preservação dos seus direitos.

A conformidade com o **RGPD** é essencial para todas as organizações que tratem dados de cidadãos da **UE**, independentemente da sua localização geográfica. A não conformidade pode resultar em sanções severas, o que tem impulsionado o desenvolvimento de ferramentas como as **CMPs**. Estas plataformas ajudam as organizações a cumprir os requisitos legais impostos por este regulamento, mas frequentemente enfrentam limitações em termos de transparência e auditabilidade.

Por fim, uma das características mais significativas do **RGPD** é que ele fornece evidências públicas e imutáveis, úteis para um **Service Provider (SP)** comprovar os acordos realizados entre um *titular de dados* e o próprio **SP** sobre os dados pessoais do titular. Esse aspecto reforça ainda mais a importância de adotar abordagens inovadoras e ferramentas tecnológicas que promovam maior confiança e conformidade com as normas regulatórias.

2.4 Blockchain e Contratos Inteligentes

Uma abordagem promissora para a auditoria de consentimentos de dados é a utilização de *blockchain* e contratos inteligentes. O *blockchain* oferece um registo distribuído e imutável que pode ser usado para documentar as interações relacionadas com o consentimento, enquanto os contratos inteligentes permitem automatizar o cumprimento das condições de consentimento estabelecidas pelos utilizadores.

A tecnologia *blockchain* distingue-se por características como descentralização, transparência, integridade e resistência à censura. Estas propriedades tornam-na particularmente adequada para garantir que os consentimentos fornecidos pelos utilizadores sejam armazenados de forma segura e auditável. Além disso, os contratos inteligentes permitem a execução automática de regras predefinidas, garantindo que os dados só sejam partilhados ou processados conforme as permissões concedidas pelo utilizador.

O conceito de *blockchain* foi inicialmente popularizado com o surgimento do Bitcoin ([Nakamoto, 2008](#)), a primeira criptomoeda descentralizada. O Bitcoin utiliza *blockchain* como um livro-razão público e imutável, onde todas as transações são registadas de forma segura e verificável sem a necessidade de uma entidade central. Com o tempo, novas aplicações da tecnologia *blockchain* emergiram, indo além das criptomoedas e incluindo domínios como gestão de identidade digital, rastreamento de cadeias de fornecimento e, mais recentemente, gestão de consentimento de dados.

Além do Bitcoin, outro marco importante na evolução da tecnologia *blockchain* foi o surgimento da Ethereum (Buterin, 2014), que introduziu a capacidade de executar contratos inteligentes. Diferente do Bitcoin, cujo foco principal é a transferência segura de valor, a Ethereum foi projetada para suportar aplicações descentralizadas através de contratos inteligentes, permitindo que regras e condições predefinidas sejam automaticamente executadas sem necessidade de intermediários. Essas capacidades tornaram a Ethereum uma das principais plataformas para aplicações *blockchain*, incluindo soluções para gestão de consentimento de dados baseadas em contratos inteligentes.

Nos últimos anos, investigadores têm proposto diferentes abordagens baseadas no uso de contratos inteligentes e na tecnologia *blockchain*, que oferecem características desejáveis, como transparência, rastreabilidade, não-repúdio e imutabilidade. Essas propostas podem ser classificadas de acordo com dois principais cenários:

- Consentimento gerido pelo titular: Quando um utilizador deseja partilhar os seus dados pessoais com terceiros, mantendo o controlo total sobre as permissões concedidas e podendo revogá-las a qualquer momento.
- Consentimento gerido por terceiros: Quando um fornecedor de serviços recolhe dados pessoais de um utilizador para posterior processamento, normalmente como parte da utilização de um produto ou serviço, sendo necessário garantir a conformidade com as regras definidas pelo utilizador e pelas regulamentações em vigor.

Atualmente, já existem várias plataformas que utilizam *blockchain* para gestão de consentimento. Algumas das principais soluções incluem:

- GDPR-Compliant Personal Data Management: Uma solução baseada em *blockchain* proposta por (Truong et al., 2020), que garante a conformidade com o RGPD através de dois sistemas de registo distribuídos. O sistema implementa o controlo de acesso através de um modelo de identidade complexa (c-ID) que combina chaves assimétricas do DS e DC, juntamente com referências encriptadas aos dados (*data_pointer*). A solução utiliza *smart contracts* específicos (*GrantConsent*, *RevokeConsent* e *DataAccess*) para regular o ciclo de vida do consentimento, mantendo um registo imutável das operações no *log_ledger* enquanto as políticas de acesso e referências aos dados são armazenadas no *3A_ledger*. Esta implementação na *Hyperledger Fabric* assegura não só o registo descentralizado do consentimento, mas também a sua validação contínua e auditável.
- Privacy by *blockchain* Design: Uma abordagem proposta por (Wirth and Kolain, 2018) que foca no registo e verificação do consentimento através da *blockchain*. O sistema utiliza *smart contracts*

especializados para gerir o ciclo de vida do consentimento, permitindo que os dados pessoais sejam armazenados *off-chain* enquanto a *blockchain* mantém apenas *hashes* e ponteiros criptográficos dos dados (*data_pointer*). A solução implementa uma arquitetura que permite que o *titular dos dados* seja notificado sempre que os seus dados são acedidos, através de um contrato inteligente que verifica a validade dos pedidos de acesso e regista todas as operações de forma transparente. Desta forma, o sistema garante que o consentimento é dado de forma específica e verificável para cada caso de uso, em vez de ser baseado em cláusulas abstratas predefinidas.

Estas soluções diferem da nossa abordagem principalmente nos seguintes aspectos: enquanto (Truong et al., 2020) implementa dois sistemas de registo separados com foco no controlo de acesso e o (Wirth and Kolain, 2018) centra-se na verificação do consentimento por caso de uso, onde, a nossa proposta foca-se na integração com uma **CMP** existente (ex. CookieChimp) e na validação em tempo real do cumprimento do consentimento.

Entre os benefícios da utilização de *blockchain* na gestão de consentimento, destacam-se:

- **Transparência e Imutabilidade:** O *blockchain* permite o registo transparente de consentimentos, garantindo que os dados não sejam alterados ou manipulados após serem armazenados.
- **Automatização:** Os contratos inteligentes podem ser configurados para permitir ou restringir o acesso aos dados com base nos consentimentos fornecidos, assegurando que as regras do **RGPD** sejam respeitadas automaticamente.
- **Auditabilidade:** Qualquer alteração nos consentimentos pode ser rastreada, permitindo a verificação da conformidade com a regulamentação e aumentando a confiança dos utilizadores.
- **Descentralização:** Ao eliminar a necessidade de intermediários para armazenar e gerir consentimentos, o *blockchain* reduz riscos de manipulação e aumenta a segurança dos dados.
- **Não Repúdio:** Uma vez registado um consentimento no *blockchain*, ele não pode ser repudiado ou modificado de forma fraudulenta, garantindo que todas as partes envolvidas possam verificar e comprovar a autenticidade do registo.

Dessa forma, o uso de *blockchain* na gestão de consentimento representa uma abordagem inovadora que pode aumentar a confiança e garantir maior transparência no tratamento de dados pessoais.

Síntese do capítulo Neste capítulo foram analisadas as principais soluções existentes para a gestão de consentimento, com destaque para as **CMPs** mais utilizadas no mercado. Verificou-se que, embora estas plataformas respondam às exigências básicas de conformidade com o **RGPD**, persistem limitações significativas relacionadas com transparência, auditabilidade e abertura do código. Foram ainda discutidas abordagens complementares, nomeadamente o recurso a tecnologias como *blockchain* e contratos inteligentes, que procuram colmatar algumas destas falhas e introduzir novos níveis de confiança e verificabilidade no tratamento de dados pessoais.

A constatação destas limitações motiva o desenvolvimento de uma solução alternativa que combine a flexibilidade das **CMPs** existentes com mecanismos robustos de transparência e auditoria. No capítulo seguinte será apresentada a arquitectura proposta, concebida de forma genérica e modular, capaz de responder a estes desafios e de servir como base para a implementação de soluções concretas de gestão de consentimento.

Capítulo 3

Visão Geral da Arquitectura

É proposto um sistema que define um fluxo de consentimento no qual cada decisão do utilizador é registada, assinada e validada, assegurando que nenhuma das partes pode manipular ou negar a informação posteriormente. Para tal, são utilizadas duas entidades principais:

- **Data Subject (DS)**: o utilizador final que interage com a interface web do serviço e fornece o consentimento através de eventos no navegador. O DS é responsável por assinar digitalmente o consentimento antes de o enviar para validação, criando uma prova verificável da sua decisão.
- **Service Provider (SP)**: o prestador de serviços que disponibiliza o website com o **CMP** à escolha e mantém o servidor de confiança. O SP recebe os consentimentos assinados pelo DS, valida a assinatura do utilizador, assina novamente o consentimento e mantém um registo imutável. Este registo permite auditoria, revogação e consulta futura.

O fluxo completo do sistema pode ser descrito de forma conceptual, independente de linguagem de programação/ferramentas utilizadas:

1. O **DS** interage com o banner de consentimento na interface web fornecida pelo **SP**.
2. A extensão do navegador do **DS** captura o evento, prepara o consentimento e assina digitalmente os dados.
3. O consentimento assinado é enviado ao servidor do **SP**, que valida a assinatura do **DS** e cria um registo final, incorporando a assinatura do **SP**.
4. O registo resultante é devolvido ao **DS**, que pode validar a assinatura do **SP**, garantindo que o consentimento foi corretamente registado e não foi alterado.
5. Ambos, **DS** e **SP**, mantêm cópias do registo, criando um histórico verificável e auditável. Posteriormente, este pode ser enviado para uma *third-party*

Desta forma, o sistema garante quatro propriedades fundamentais:

- **Transparência:** todos os passos do processo podem ser verificados pelo utilizador ou pelo prestador de serviços.
- **Autenticidade e Integridade:** assinaturas digitais asseguram que os consentimentos não foram alterados e que provêm das entidades corretas.
- **Não repúdio:** o **DS** não pode negar a sua decisão, e o **SP** não pode alegar que não recebeu ou validou o consentimento.
- **Auditabilidade:** o histórico de consentimentos permanece acessível a ambas as entidades, permitindo conformidade com requisitos legais e regulação de proteção de dados.

3.1 Componentes

O sistema é constituído por três componentes principais: a **interface web**, a **extensão do navegador** e o **servidor**. A interface web representa o ponto de contacto direto com o utilizador e disponibiliza o *banner* de consentimento. A extensão do navegador actua como intermediário, recebendo os eventos provenientes da interface web e tratando do processo de assinatura digital. Por fim, o servidor desempenha o papel de entidade de confiança, responsável por validar as assinaturas recebidas e criar um registo imutável com os consentimentos.

3.2 Interação Entre Componentes

Extensão ↔ Interface Web

A extensão comunica com as interações do cliente com a interface web:

- Deteta ações do utilizador no *banner* de consentimento
- Inicia processamento criptográfico do consentimento no background
- Confirmação de sucesso devolvida ao utilizador

Extensão ↔ Servidor

A extensão estabelece um canal com o servidor:

- Importação dos certificados e chave privada

- Obtenção do certificado do servidor
- Envia o consentimento assinado como também o certificado do cliente
- Recebe registo do consentimento com assinatura do cliente, servidor e valida a última

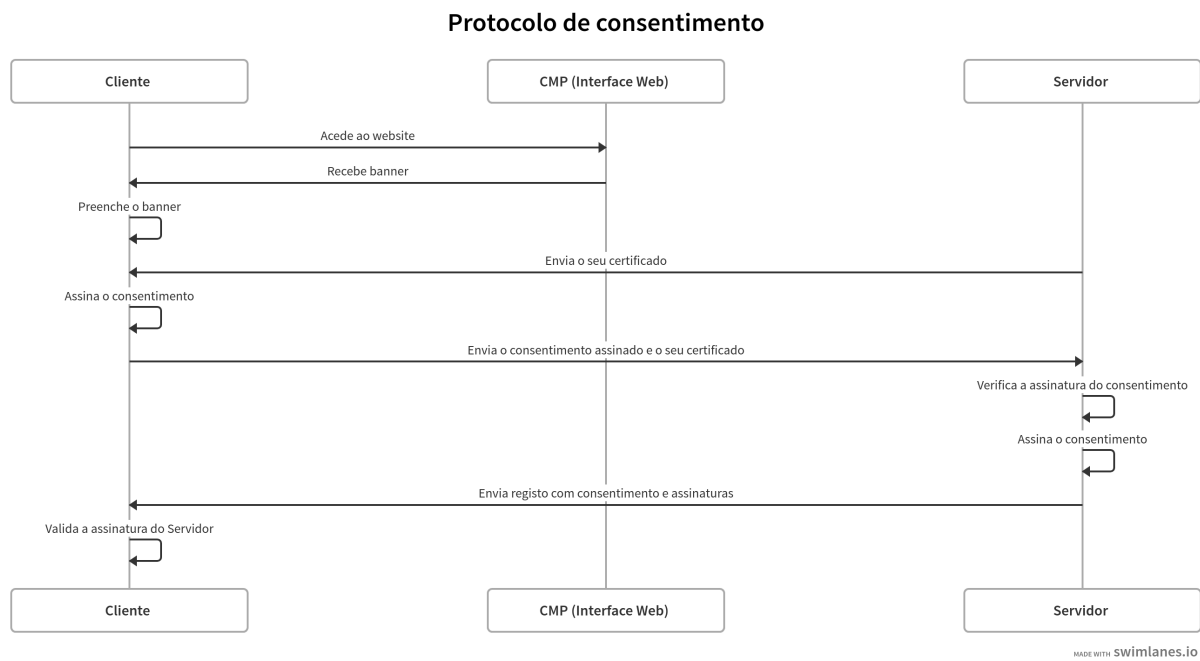


Figura 4: Diagrama do protocolo

3.3 Lógica de Funcionamento

A estratégia proposta assenta numa abordagem de assinaturas digitais, na qual o cliente e servidor participam ativamente na criação de um registo de consentimento verificável e imutável. Esta abordagem garante quatro propriedades fundamentais: **transparência**, na medida em que todos os passos podem ser verificados; **descentralização**, dado que nenhuma das partes detém controlo unilateral; **não-repúdio**, assegurando que os consentimentos prestados não podem ser posteriormente negados; e **auditabilidade**, uma vez que o histórico completo permanece acessível a ambas as entidades.

3.4 Modelo de confiança

O processo de recolha e gestão de consentimentos digitais enfrenta vários desafios de confiança. Em particular, surgem questões relacionadas com a falta de garantias sobre a identidade das entidades envolvidas e a ausência de mecanismos claros de auditoria. Estes problemas levantam dúvidas tanto para os **DS**, que necessitam de garantias de transparência e controlo sobre as suas decisões, como para os **SP**, que precisam de provas fiáveis para demonstrar conformidade regulatória.

Para colmatar estas lacunas, o modelo de confiança proposto assenta em duas camadas principais de proteção: **certificados digitais**, que funcionam como método de autenticação das entidades envolvidas, e **assinaturas digitais**, que asseguram a autenticidade, a integridade e o não repúdio das decisões de consentimento. Graças a estes mecanismos, cada interação é não só verificável por ambas as partes, como também **auditável**, permitindo a reconstrução fiel do histórico de consentimentos e reforçando a conformidade com requisitos legais como o **RGPD**.

3.4.1 Certificados digitais

Os certificados desempenham um papel fundamental na garantia de identidade e na criação de comunicações seguras. De forma simplificada, um certificado digital é um ficheiro que associa uma chave pública a uma entidade (por exemplo, um utilizador ou um servidor). Esta associação é validada por uma **Certificate Authority (CA)**, que funciona como uma entidade de confiança responsável por emitir e assinar certificados.

A chave privada deve permanecer confidencial, pois é utilizada para operações críticas, como a criação de assinaturas digitais. Já a chave pública, incluída no certificado, pode ser partilhada e serve para validar essas assinaturas.

A **CA** raiz (root **CA**) é a autoridade de topo, responsável por assinar certificados de entidades inter-médias ou diretamente de clientes e servidores, como é o caso. Este mecanismo hierárquico garante que, ao receber um certificado, é possível verificar a sua autenticidade através da cadeia de confiança estabelecida pela autoridade certificadora.

3.4.2 Assinatura Digital

A assinatura digital é um mecanismo criptográfico baseado em algoritmos de chave assimétrica, concebido para garantir a autenticidade e a integridade de uma mensagem ou documento eletrónico. O seu funcionamento baseia-se em dois elementos fundamentais: a chave privada e a chave pública. A chave privada é utilizada para gerar a assinatura digital e deve permanecer secreta, acessível apenas ao titular. A chave pública é distribuída, neste caso através de certificados digitais, permitindo que qualquer entidade verifique a validade da assinatura.

Desta forma, a assinatura digital assegura três propriedades essenciais (Subramanya and Yi, 2006):

- **Autenticidade:** confirma que a mensagem foi assinada pela entidade detentora da chave privada correspondente.
- **Integridade:** garante que o conteúdo não foi alterado após a assinatura.
- **Não repúdio:** impede que o autor negue a sua participação no processo, tornando a assinatura uma evidência legalmente relevante.

As assinaturas digitais constituem o fundamento de sistemas confiáveis de registo de consentimentos, transações financeiras e documentos eletrónicos, sendo amplamente utilizadas em padrões de segurança e infraestruturas de chave pública.

3.5 Registo do consentimento

O consentimento do utilizador é preferencialmente um registo estruturado que pode ser interpretado e processado de forma padronizada. Este registo deve ser interoperável, auditável e verificável, permitindo que diferentes sistemas o leiam e validem sem ambiguidade.

Para garantir estas propriedades, o consentimento é assinado digitalmente tanto pelo utilizador como pela entidade que o recebe. A troca de certificados entre as partes possibilita a verificação mútua das assinaturas, reforçando a confiança no processo. O objecto resultante agrega informação relevante so-

bre as decisões do utilizador, bem como metadados necessários à validação, mantendo a integridade e autenticidade do consentimento.

Adotar um padrão estruturado para o consentimento permite:

- Facilitar a integração com diferentes sistemas/aplicações;
- Manter um registo auditável e verificável ao longo do tempo;

3.6 Benefícios da Arquitectura

A arquitectura proposta apresenta benefícios distintos para os diferentes intervenientes. Do ponto de vista do utilizador, a solução facilita aceder aos consentimentos prestados mais facilmente, assegurando transparência no processo. Adicionalmente, o utilizador tem a possibilidade de verificar a integridade dos registos, garantindo que não foram alvo de manipulação, e dispõe ainda de mecanismos que lhe conferem autonomia para comprovar eventuais incumprimentos por parte do prestador de serviços.

Para as organizações, a arquitectura disponibiliza provas fiáveis de consentimentos válidos, que podem ser utilizadas em auditorias ou processos de verificação de conformidade. Deste modo, contribui para a redução dos riscos associados ao incumprimento das regulamentações em vigor, promovendo maior confiança e segurança jurídica no tratamento de dados pessoais.

Síntese do capítulo Neste capítulo foi apresentada a arquitectura conceptual da solução proposta, destacando os seus principais componentes, a lógica de funcionamento e os mecanismos criptográficos utilizados, como certificados digitais e assinaturas digitais. Foi ainda detalhado o modo como o consentimento é representado e registado, assegurando propriedades fundamentais como autenticidade, integridade, não repúdio, transparência e auditabilidade. Esta arquitectura foi concebida de forma genérica e modular, de modo a poder ser aplicada em diferentes contextos, independentemente das tecnologias concretas utilizadas.

No capítulo seguinte será descrita a implementação prática desta arquitectura, materializada numa solução funcional que integra os princípios aqui definidos. Essa implementação permitirá demonstrar a viabilidade do modelo proposto, validando o seu funcionamento e evidenciando os benefícios alcançados face às limitações identificadas no estado da arte.

Capítulo 4

Implementação da solução

A implementação da solução envolveu a escolha das tecnologias mais adequadas para cada uma das entidades faladas anteriormente. O *back-end* do servidor foi desenvolvido em *Node.js*, enquanto o website é constituído por uma página em *HTML* simples, integrando um *snippet* do *script* de um **CMP** Open-Source (*Klaro.js*) para a apresentação do *banner* de consentimento. Do lado do cliente, foi implementada uma extensão de navegador em JavaScript, responsável pela captura das interações do utilizador e pela execução das operações criptográficas necessárias. Embora se trate de uma extensão em JavaScript nativo, recorreu-se ao Vue.js como *framework*, em conjunto com o Vite como *bundler*, de modo a permitir a integração de bibliotecas externas, nomeadamente o *node-forge* para o tratamento de chaves e certificados. Adicionalmente, o processo de criação e gestão de certificados digitais foi realizado com recurso ao *OpenSSL*, garantindo a relação de confiança entre cliente e servidor. O fluxo de comunicação resultante contempla as etapas de recolha, assinatura, envio, validação e registo do consentimento em ambas as entidades, assegurando a autenticidade, integridade e verificabilidade do processo.

4.1 Open-Source

A recolha de consentimentos do lado do servidor apresenta-se como um desafio, tanto em termos de arquitetura como de conformidade regulatória. Este desafio é amplificado pelo facto de que a maioria das soluções disponíveis de **CMPs** são soluções fechadas, dificultando a transparência e assim a auditabilidade dos processos.

As soluções fechadas limitam a capacidade de compreender como os dados são processados e armazenados após o consentimento, dificultando a verificação independente de que as escolhas do utilizador estão a ser respeitadas. Esta falta de visibilidade cria barreiras à adoção de práticas verdadeiramente transparentes e auditáveis, essenciais para garantir a conformidade com regulamentações como o **RGPD**.

Para alcançar este objetivo sem a necessidade da criação de um **CMP**, a solução adota uma abor-

dagem que combina transparência, auditabilidade e eficiência com o uso de um **CMP** Open-source. A transparência deve ser garantida, permitindo aos utilizadores compreender exatamente como os seus dados estão a ser processados. Idealmente esta mesma interface deve permitir uma visão clara de como os dados são armazenados e utilizados, assegurando que o utilizador tem algum controlo sobre as suas escolhas, pois caso seja evidenciado que as suas escolhas não são cumpridas este mesmo utilizador tem às suas mãos a ferramenta necessária para o provar e agir sobre essa falha.

Com estas características, a solução não pretende apenas abordar os desafios técnicos e regulatórios, mas também promover confiança, transparência e conformidade no tratamento de dados pessoais.

4.1.1 A Necessidade de **CMPs** Open-Source

Uma das principais motivações deste trabalho é a necessidade de soluções mais transparentes e auditáveis. As **CMPs** tradicionais, muitas das quais são soluções fechadas, não permitem aos utilizadores ou aos investigadores uma verificação independente de como os dados são tratados. Por outro lado, **CMPs** Open-source oferecem a vantagem de serem transparentes e acessíveis, permitindo que o código-fonte seja inspecionado e auditado.

Plataformas Open-source como o **Klaro.js** oferecem uma maior transparência, permitindo que os desenvolvedores compreendam melhor como os dados são processados e oferecendo a possibilidade de auditar o processo de consentimento. Através do acesso ao código-fonte, os utilizadores e empresas podem verificar se a implementação está em conformidade com as regulamentações de privacidade e garantir que o consentimento dos utilizadores é gerido de forma adequada.

4.2 Servidor e Website

Para capturar a interação do **DS** é disponibilizado um website, este website trata-se de um site estático feito em HTML no qual chamamos o *script* de KlaroJS integrado no nosso HTML, como é comum na integração de **CMPs**. Podemos definir quais são os nossos serviços em um ficheiro chamado `config.js` que deve estar estruturado como o exemplo disponibilizado pelo [Klaro \(2025\)](#). Neste caso, como não têm peso quais os serviços a ser utilizados, foram mantidos os valores defaults.

Podemos ver aqui um exemplo de um serviço:

```
1 // This is a list of third-party services that Klaro will manage for you.
2 services: [
```

```

3  {
4      name: 'twitter',
5      default: false,
6      contextualConsentOnly: true,
7      purposes: ['marketing'],
8  },

```

Este ficheiro é depois chamado no referido script anteriormente `klaro.js`, que se trata de uma compilação de tudo que compõe o **CMP** KlaroJS. Aqui, é possível chamar o *script* através de um *endpoint* exposto pelo o próprio desenvolvedor da ferramenta, ou então importar para o projeto. Nesta solução, foi escolhido o segundo método para a utilização da nossa própria configuração. Sendo assim, na nossa página, só foi necessário chamar estes dois *scripts* para implementar o serviço.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Consent Management POC</title>
6
7      <script defer type="text/javascript" src="config.js"></script>
8      <script defer type="text/javascript" src="klaro.js"></script>
9  </head>
10 ...

```

Com isto, tem-se as ferramentas necessárias para ter um ponto de interação com o **DS**.

Toda a lógica de troca de consentimento e de certificados é suportada por um servidor *back-end*, desenvolvido em NodeJS, que expõe dois *endpoints* principais para comunicação com o cliente.

O primeiro *endpoint*, acessível através de um pedido GET em `/api/server_certificate`, devolve o certificado público do servidor. Este certificado é utilizado pelo cliente para validar as comunicações e garantir a autenticidade das assinaturas digitais. Em caso de sucesso, o certificado é devolvido em formato JSON, caso contrário, o servidor responde com o respetivo erro.

O segundo *endpoint*, acessível através de um pedido POST em `/api/consent`, recebe do cliente um consentimento assinado no formato JWS. O corpo da requisição inclui a chave pública do cliente, o consentimento propriamente dito e a assinatura associada. O servidor procede então a várias operações:

1. Verificação da assinatura do cliente — utilizando a chave pública fornecida através do certificado, confirma-se que o consentimento foi efetivamente assinado pelo cliente, assegurando a integridade e autenticidade da informação recebida.

2. Criação de um objeto JWS assinado pelo servidor e cliente — após validação, o consentimento é encapsulado na estrutura JWS, assinado com a chave privada do servidor, de modo a gerar evidência verificável para ambas as partes.
3. Resposta ao cliente — é devolvido ao cliente um objeto em formato JSON, contendo o JWS assinado pelo cliente e servidor e uma mensagem de confirmação. Em caso de falha, é emitida uma resposta de erro.

Deste modo, o servidor é responsável por atuar como entidade de confiança, responsável por validar as mensagens recebidas e por emitir, como resposta, um JWS assinado com a sua chave privada. Esse JWS não constitui, por si só, a conclusão do processo: é posteriormente validado pelo cliente com recurso ao certificado público obtido em `/api/server_certificate`, confirmando a autenticidade da assinatura do servidor e a integridade/consistência do *payload* antes de proceder ao registo local do consentimento.

4.3 JWS

No âmbito da gestão do consentimento, a organização e validação dos mesmos exigem um formato seguro, interoperável e verificável. Entre várias alternativas, optou-se por utilizar o *JSON Web Signature* (JWS), conforme definido no RFC 7515 ([Jones et al., 2015](#)).

4.3.1 Descrição Geral (Baseado no RFC 7515)

De acordo com o RFC 7515, um JWS representa conteúdos protegidos com assinaturas digitais ou códigos de autenticação de mensagem (MAC), usando estruturas de dados em JSON.

Existem duas formas de serialização definidas:

- **Compact Serialization:** uma representação mais concisa, adequada a ambientes com restrições de espaço, como cabeçalhos HTTP ou parâmetros de URL.
- **JSON Serialization:** uma representação em JSON que permite múltiplas assinaturas ou MACs sobre o mesmo conteúdo, com maior clareza e flexibilidade.

O JWS baseia-se em três componentes principais:

1. **Header:** contém metadados como o algoritmo de assinatura (por exemplo, PS256), o tipo de objeto (por exemplo, JWT), e possivelmente outros parâmetros relevantes.

2. **Payload:** o conteúdo a assinar (neste caso, os detalhes do consentimento), que é codificado em Base64URL.
3. **Signature:** o resultado da assinatura digital aplicada ao *header* e *payload*, garantindo integridade e autenticidade.

4.3.2 Aplicação ao Caso de Consentimento

No sistema implementado, o consentimento do utilizador é representado através de um JWS, com estas características específicas:

- O **payload** inclui informação relevante sobre o consentimento (por exemplo, serviços aceites/rejeitados).
- O **header** utiliza o algoritmo PS256, que combina RSASSA-PSS com SHA-256.
- São produzidas pelo menos duas assinaturas:
 - Uma assinatura gerada pelo cliente (extensão do navegador), garantindo que foi o utilizador quem autorizou o consentimento.
 - Outra assinatura adicional do servidor, garantindo que o servidor valida e “assina” o consentimento final, reforçando a confiabilidade e verificabilidade.
- Estas múltiplas assinaturas tornam o esquema equivalente à *JSON Serialization* do JWS, que suporta várias assinaturas sobre o mesmo *payload*.

4.3.3 Vantagens neste contexto

- **Integridade e Autenticidade:** Qualquer modificação no *payload* ou nas assinaturas invalida o JWS, protegendo contra manipulação.
- **Não repúdio:** A assinatura do cliente impede que este negue ter dado o consentimento, sendo uma evidência legalmente relevante.
- **Verificabilidade por ambas as partes:** Cliente e servidor conseguem validar, de forma independente, que o consentimento é autêntico e válido.
- **Compatível com padrões e interoperável:** Facilita futuras integrações com outras ferramentas.

4.4 Extensão no Cliente

Do lado do cliente desenvolveu-se uma extensão para o navegador, responsável pela gestão da troca de consentimentos. Esta foi implementada em JavaScript ([Mozilla, 2025](#)). Inicialmente foi utilizado o `manifest v2`, dado que os testes foram realizados em *Firefox*. Mais tarde, procedeu-se à migração para o `manifest v3`, o que se revelou um processo simples e sem complicações.

Module Bundlers

Uma vez que se trata de uma aplicação em JavaScript, não é viável utilizar apenas um ficheiro de JavaScript puro para integrar outras bibliotecas. Para isso, é necessário recorrer a *module bundlers*. Neste caso, como já foi referido acima, foi utilizado o Vite (ferramenta padrão do Vue.js), uma solução *lightweight* que satisfaz esta necessidade. Dado que apenas JavaScript nativo não é totalmente compatível com todos os navegadores, este *bundler* permite gerar o *build* do nosso *source code*, bem como das *third-party dependencies*, de forma recursiva. As próprias *dependencies* podem ter outras *dependencies*, como no caso da biblioteca `node-forge`, permitindo consolidar todos os ficheiros num único output.

O desenvolvimento web moderno depende fortemente de *module bundlers*. Estas ferramentas processam o código-fonte da aplicação e as suas *dependencies*, produzindo *assets* otimizados e prontos para *deploy*. Ao contrário das linguagens compiladas, como C ou Go, que produzem um único executável binário, o JavaScript é interpretado dinamicamente no *browser*. Isto cria desafios de compatibilidade e *performance*, que os *bundlers* ajudam a resolver.

Propósito e Funcionalidade

As aplicações JavaScript são frequentemente compostas por múltiplos módulos, importados utilizando o `import` ou `require`. *Module bundlers*, como o Vite (usado pelo Vue por omissão), consolidam estes módulos num ou poucos ficheiros. Durante este processo, podem também:

- Fazer a transpilação de *Modern JavaScript* ou *TypeScript* para garantir compatibilidade com *browsers* mais antigos;
- *Minify code*, reduzir o tamanho dos ficheiros para carregamento mais rápido;
- *Tree-shaking*, remover código não utilizado;
- *Bundle static assets*, como *CSS*, imagens e *fonts*.

Diretório dist

Após o *bundling*, por convenção, os ficheiros processados são normalmente colocados na pasta *dist* (*distribution*). Esta pasta contém *assets* prontos para produção, totalmente compilados e otimizados. Permite separar o código-fonte de desenvolvimento do *output* final para *deploy*, garantindo consistência e eficiência.

4.5 Gestão de Certificados

Para a criação dos certificados, foi usada a ferramenta do OpenSSL para testes locais. Para obter estes mesmos certificados por parte do servidor e cliente, procedemos à criação de uma *rootCA*. Sendo assim, neste caso, ambas utilizam a mesma *root CA*. Com isto, precisamos que cada uma das entidades tenha a sua chave privada.

Criação da *rootCA*:

```
1 CANAME=Uminho-RootCA
2 openssl genrsa -out $CANAME.key 2048
3 openssl req -x509 -new -nodes -key $CANAME.key -sha256 -days 1826 -out
  $CANAME.crt
```

Com isto, procedemos à criação das chaves privadas e *certification requests* de ambas as entidades.

```
1 openssl genrsa -out {server/client}.key 2048
2 openssl req -new -key {server/client}.key -out {server/client}.csr
3 openssl x509 -req -in {server/client}.csr -CA ca/rootCA.crt -CAkey ca/
  rootCA.key -CAcreateserial -out {server/client}.crt -days 825 -sha256
```

Obtemos assim os certificados de ambas as entidades.

No entanto, a extensão necessita ainda de aceder ao certificado e à chave privada do cliente (*client.crt* e *client.key*). Estes são armazenados no *local storage* do navegador, sendo depois carregados pela extensão:

```
1 ...
2 certPEM = localStorage.getItem("cert");
3 certPEM = this.formatPem(certPEM, "CERTIFICATE");
4
5 privKey = localStorage.getItem("privKey");
6 privKey = this.formatPem(privKey, "PRIVATE KEY");
7 ...
```

A integração com o módulo `node-forge` foi fundamental para o manuseamento de chaves e certificados. Mas como em extensões de navegador apenas é permitido código JavaScript nativo, foi necessário proceder à sua compilação e empacotamento. Para tal, recorreu-se a um *bundler*, tendo sido escolhida a *framework* VueJS (Vue, 2025).

Do lado do servidor, a gestão de chaves e certificados é simplificada através do recurso ao `node-forge`, tal como no cliente. O servidor realiza o carregamento das chaves RSA e do certificado diretamente a partir do sistema de ficheiros, extraíndo os elementos necessários para validação e assinatura dos consentimentos.

O seguinte *snippet* demonstra a lógica implementada:

```
1 loadRSASigningKeys() {
2   const certPem = fs.readFileSync('server.crt', 'utf8');
3   const cert = forge.pki.certificateFromPem(certPem);
4   const publicKey = forge.pki.publicKeyToPem(cert.publicKey);
5
6   const privateKey = fs.readFileSync('server.key', 'utf8');
7
8   this.serverKeys.rsaPrivateSigningKey = privateKey;
9   this.serverKeys.rsaPublicSigningKey = publicKey;
10  this.serverCert = certPem;
11 }
```

Neste processo:

- O certificado do servidor é lido e decodificado em formato PEM, permitindo extrair a chave pública para validação das assinaturas.
- A chave privada é carregada diretamente do ficheiro correspondente, sendo utilizada para assinar os JWS recebidos do cliente.
- O `node-forge` facilita a manipulação de certificados e a conversão entre formatos PEM e objetos manipuláveis em JavaScript.

4.6 Fluxo de Consentimento

O processo inicia-se quando o utilizador interage com o *banner* de consentimento (aceitação ou rejeição), disponibilizado pelo *Klaro.js* (Figura 5).

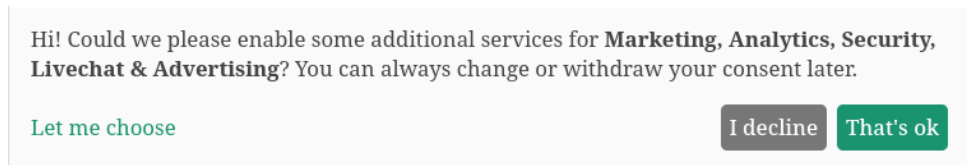


Figura 5: Banner padrão do Klaro.js

A extensão do navegador captura este evento através de um *listener*, que desencadeia a função principal `processConsent`. Este processo caracteriza-se pelas seguintes etapas:

1. Obtenção do certificado público do servidor, essencial para validar a autenticidade das mensagens recebidas.
2. Carregamento das chaves do cliente (`client.key` e `client.crt`) do *local storage*.
3. Assinatura digital do consentimento pelo cliente, criando uma evidência verificável.
4. Envio do consentimento assinado ao servidor.
5. Validação do consentimento no servidor, incluindo a verificação da assinatura do cliente.
6. Criação de um JWS assinado pelo servidor e envio de resposta ao cliente.
7. Validação final do JWS pelo cliente, assegurando a integridade e autenticidade do *payload* antes de registrar localmente o consentimento.

Este fluxo garante que todas as interações são auditáveis, permitindo rastreabilidade e conformidade com requisitos legais de proteção de dados.

O fluxo descrito pode ser visualizado no diagrama da Figura 6.

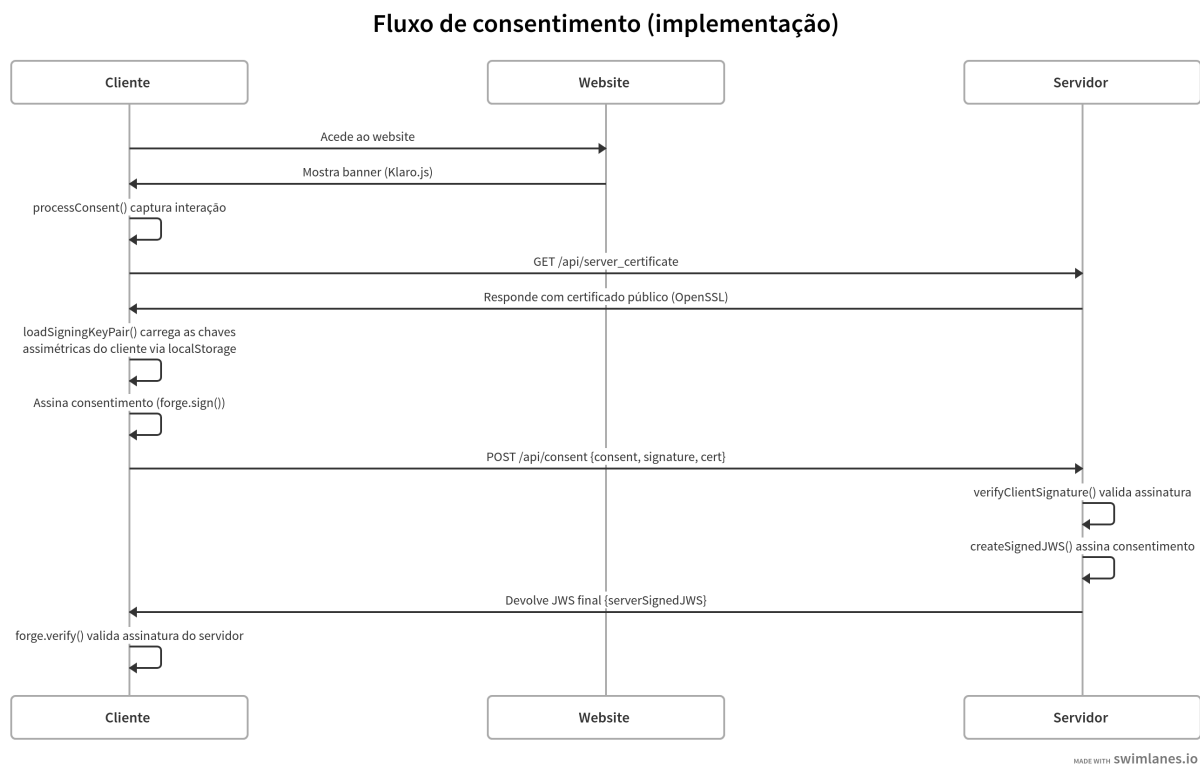


Figura 6: Diagrama do protocolo

4.7 Registo do consentimento

O registo com o consentimento e as assinaturas é o resultado final do fluxo implementado: um registo digital que comprova a aceitação ou rejeição pelo utilizador de determinados serviços ou finalidades. Este registo é implementado como um *JSON Web Signature (JWS)*, que encapsula:

- O *payload* contendo os detalhes do consentimento do utilizador.
- A assinatura digital do cliente, garantindo que foi realmente o utilizador a autorizar.
- A assinatura digital do servidor, confirmando a validação e integridade do consentimento.

Ambas as entidades (cliente e servidor) mantêm este JWS, permitindo consultas futuras, auditoria e eventual revogação do consentimento. O JWS assegura:

- Autenticidade — a assinatura do cliente comprova que o consentimento foi emitido pelo utilizador correto.
- Integridade — alterações no *payload* invalidam as assinaturas, evitando manipulação.

- Não repúdio — o cliente não pode negar a sua decisão, uma vez que a assinatura digital é inequívoca.

O JWS resultante contém dois elementos principais:

- **Payload:** o *payload* codificado em *base64* com os dados do consentimento do utilizador.
- **Signatures:** um array com uma ou mais assinaturas digitais. No nosso caso, inclui a assinatura do cliente e a assinatura do servidor.

Um exemplo de JWS gerado é o seguinte:

```

1 {
2   "payload": "eyJjb25zZW50cyI6eyJ0d2l0dGVyIjpmYWxzZSw...",
3   "signatures": [
4     {
5       "header": {"typ": "JWT", "alg": "PS256"},
6       "signature": "b1Xn5AaxYZWZNfHoeL-SWTAYsbt8yFWjJiPTK_rlIoPwTdukp9wpn
          ... "
7     },
8     {
9       "header": {"typ": "JWT", "alg": "PS256"},
10      "signature": "BPVz73atRIFhzRx6YVSHW0kEX6Rb-
          hL0Xoah0c2uxX9EPDs5RSVvYuNzpoX_Vv..."
11    }
12  ]
13 }

```

Esta estrutura assegura que tanto o cliente como o servidor possuem uma prova verificável do consentimento, permitindo auditoria, revogação ou consulta futura, garantindo a conformidade com requisitos definidos na criação deste POC (Proof of Concept). Desta forma, o mecanismo fornece uma solução robusta, segura e transparente para a gestão de consentimentos no contexto do sistema implementado.

Síntese do capítulo Neste capítulo foi descrita a implementação prática da solução proposta, detalhando as tecnologias seleccionadas e a forma como foram integradas para concretizar a arquitectura conceptual apresentada anteriormente. Foram discutidos os diferentes componentes cliente, servidor e interface web, bem como o papel de elementos fundamentais como certificados digitais, assinaturas digitais e o formato JWS na criação de registos de consentimento auditáveis e verificáveis. Esta implementação demonstrou a viabilidade técnica da proposta e a sua adequação a um ambiente real de gestão de consentimento.

No capítulo seguinte são apresentadas as conclusões finais deste trabalho, destacando os principais contributos alcançados e a forma como estes respondem às limitações identificadas no trabalho relacionado. Serão igualmente discutidas as perspectivas de evolução futura, apontando caminhos para o aprofundamento e expansão da solução aqui desenvolvida.

Capítulo 5

Conclusões e trabalho futuro

5.1 Conclusões

O principal objetivo definido para este trabalho consistia em conceber e prototipar um sistema de gestão de consentimento que, para além de recolher as escolhas do utilizador, permitisse também auditar o processo de forma transparente, verificável e em conformidade com o **RGPD**. Para tal, partiu-se da análise crítica das limitações das **CMPs** tradicionais, em particular a falta de transparência e de mecanismos de auditoria, e delineou-se uma solução que permite acrescentar estas garantias independentemente da plataforma utilizada. A proposta integra mecanismos criptográficos robustos que podem ser aplicados em qualquer CMP existente, potenciando a transparência e a verificabilidade do processo de gestão de consentimento.

Este objetivo foi alcançado através da definição de uma arquitetura genérica baseada em três entidades principais (utilizador, extensão no navegador e servidor), capaz de registar consentimentos com recurso a assinaturas digitais e certificados. A implementação prática validou a viabilidade deste modelo, recorrendo a tecnologias como Node.js no servidor, Klaro.js para o banner de consentimento e uma extensão de navegador em JavaScript, suportada por bibliotecas de criptografia. O processo resultou num fluxo completo de recolha, assinatura, validação e registo de consentimentos em formato JWS, garantindo autenticidade, integridade, não repúdio e auditabilidade.

Desta forma, pode afirmar-se que os objetivos delineados foram cumpridos. A solução concebida não só demonstra ser possível conjugar simplicidade de utilização com garantias fortes de segurança e confiança, como também responde diretamente ao principal problema identificado: a ausência de mecanismos que permitam ao utilizador salvaguardar-se em caso de não conformidade. Com o registo duplamente assinado em formato JWS, o utilizador dispõe de uma prova verificável do consentimento que efetivamente forneceu, podendo assim contestar eventuais falhas do lado do prestador de serviços. Este mecanismo de auditoria constitui o contributo mais relevante desta dissertação, ao assegurar que tanto

utilizador como organização partilham um histórico comum, transparente e verificável.

5.2 Trabalho futuro

Embora a solução apresentada tenha demonstrado a viabilidade de integrar mecanismos de auditoria e verificação de consentimentos em plataformas existentes, permanecem diversas oportunidades para evolução e aprofundamento do trabalho. Esta secção discute possíveis direções para trabalhos futuros, destacando melhorias técnicas e expansão de funcionalidades que possam aumentar a transparência, a confiabilidade e a escalabilidade do sistema. O objetivo é fornecer uma perspetiva sobre como a investigação presente pode ser prolongada, contribuindo para soluções mais robustas e abrangentes no domínio da gestão de consentimento de dados.

5.2.1 Complementação do fluxo com blockchain e contratos inteligentes

A prova de conceito desenvolvida assegura a autenticidade, integridade e verificabilidade dos consentimentos através de assinaturas digitais e certificados, no entanto, uma possível extensão futura seria a integração com um *ledger* público.

A utilização desta tecnologia permitiria reforçar a imutabilidade dos registos, uma vez que o consentimento poderia ser armazenado de forma distribuída numa rede pública, resistente a manipulações. Neste cenário, após o consentimento ser validado e assinado por ambas as partes, o servidor poderia proceder ao registo do identificador único associado ao consentimento (ID) numa *blockchain* pública, como a **Ethereum**.

Com este mecanismo, seria possível garantir que qualquer entidade interessada — incluindo o próprio utilizador — pudesse verificar, de forma independente, a existência e consistência dos consentimentos registados. O registo em *blockchain* funcionaria, assim, como uma prova adicional de confiança, complementando as garantias já oferecidas pelo sistema atual.

Além disso, esta abordagem abriria caminho para auditorias independentes e mecanismos automatizados de verificação, assegurando que os consentimentos mantêm-se inalterados ao longo do tempo e promovendo uma maior transparência na gestão de dados pessoais.

A exploração desta integração com *blockchain* e contratos inteligentes constitui, portanto, um rumo relevante para trabalho futuro, potenciando a robustez e a confiança da solução aqui apresentada.

5.2.2 Integração com keychain do sistema do cliente

Outra direção futura consiste em explorar a integração da extensão com a importação da *keychain* do sistema operativo do utilizador. Esta integração permitiria gerir de forma mais segura as chaves privadas utilizadas na assinatura dos consentimentos, reduzindo o risco de exposição ou uso indevido. Além disso, garantiria maior comodidade para o utilizador, que não precisaria de gerir manualmente chaves criptográficas nem depender de soluções externas para armazenar credenciais sensíveis.

5.2.3 Disponibilização da solução como projeto Open-source

Um objetivo adicional é tornar a solução disponível como projeto Open-source. Esta abordagem facilitaria auditorias independentes, fomentaria a confiança por parte dos utilizadores e permitiria que a comunidade contribuísse para melhorias, correções de segurança e evolução da plataforma. A abertura do código reforçaria ainda a transparência e a auditabilidade do sistema, promovendo uma adoção mais ampla em contextos académicos, corporativos e de investigação.

Capítulo 6

Avaliação

Este capítulo apresenta a avaliação da solução implementada, tanto em termos funcionais como de desempenho. O objetivo é demonstrar a viabilidade da arquitetura proposta, analisando a sua usabilidade, a robustez dos mecanismos criptográficos e desempenho.

6.1 Avaliação Funcional

Para validar o correto funcionamento da solução, foram realizados testes que cobrem o fluxo completo de consentimento:

1. Apresentação do *banner* de consentimento através do Klaro.js.
2. Captura da interação do utilizador pela extensão no cliente.
3. Assinatura digital do consentimento no cliente.
4. Envio do consentimento assinado para o servidor.
5. Validação da assinatura do cliente no servidor.
6. Criação de um JWS com assinaturas do cliente e servidor.
7. Validação final pelo cliente e registo local do consentimento.

A Figura 5 já ilustrou a interface do banner apresentada ao utilizador. De seguida, mostram-se exemplos de payloads e respetivos JWS gerados pelo sistema.

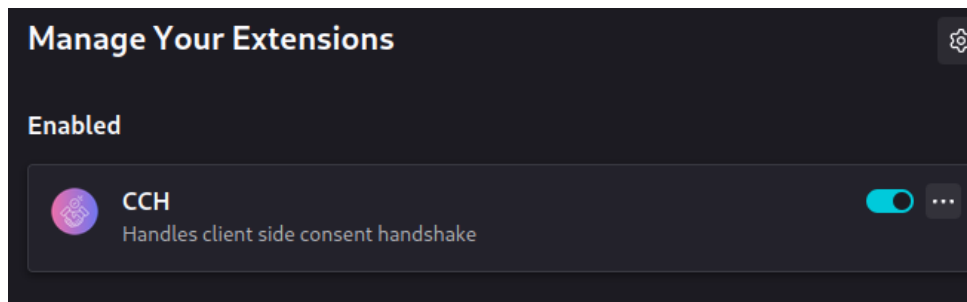


Figura 7: Extensão carregada no browser

Exemplo 1 – Recusar todos os consentimentos

```

1 {
2   consents: {
3     twitter: false,
4     youtube: false,
5     matomo: false,
6     inlineTracker: false,
7     externalTracker: false,
8     intercom: false,
9     mouseflow: false,
10    adsense: false,
11    camera: false,
12    cloudflare: true
13  },
14  confirmed: false,
15  timestamp: '2025-10-01T17:14:47.863Z'
16 }

```

Listing 6.1: Payload com rejeição de todos os consentimentos.

```

1 {
2   "payload": "eyJjb...",
3   "signatures": [
4     {
5       "header": { "typ": "JWT", "alg": "PS256" }, "signature": "g_E073..."
6     },
7     {
8       "header": { "typ": "JWT", "alg": "PS256" }, "signature": "iLcMTs..."
9     }
10  ]

```

```
11 }
```

Listing 6.2: Exemplo de JWS gerado para recusar todos os consentimentos.

Podemos ver aqui as principais estruturas de dados que são trocadas entre o cliente e servidor.

Exemplo 2 – Aceitar todos os consentimentos

```
1 {
2   consents: {
3     twitter: true,
4     youtube: true,
5     matomo: true,
6     inlineTracker: true,
7     externalTracker: true,
8     intercom: true,
9     mouseflow: true,
10    adsense: true,
11    camera: true,
12    cloudflare: true
13  },
14  confirmed: true,
15  timestamp: '2025-10-01T17:15:33.744Z'
16 }
```

Listing 6.3: Payload com aceitação de todos os consentimentos.

```
1 {
2   "payload": "eyJjb... ",
3   "signatures": [
4     {
5       "header": { "typ": "JWT", "alg": "PS256" },
6       "signature": "n0VOK..."
7     },
8     {
9       "header": { "typ": "JWT", "alg": "PS256" }, "signature": "f2lBlY..."
10    }
11  ]
12 }
```

Listing 6.4: Exemplo de JWS gerado para aceitação de todos os consentimentos.

No caso da aceitação dos consentimentos, a funcionalidade e a estrutura dos dados não se altera significativamente.

Exemplo 3 — Subconjunto de consentimentos (5 serviços)

Foram também realizados testes com *payloads* reduzidos, contendo apenas 5 serviços. Os resultados confirmam que, consoante for menor o *payload*, melhor são os tempos de verificação e criação dos JWS.

6.2 Avaliação de Desempenho

Foram conduzidos testes de medições de desempenho para avaliar a eficiência da solução. Os tempos de execução foram registados, tanto para a criação do JWS, como para a verificação da assinatura. A Tabela 2 resume os resultados obtidos. É necessário ter em conta que, na própria criação do JWS é feita ainda a assinatura do servidor sobre o consentimento. E a verificação da assinatura do cliente é efetuada pelo servidor.

Tabela 2: Resultados de desempenho em diferentes cenários de payload.

Cenário	Tamanho do payload	Tempo verificação	Tempo criação JWS
Recusa total	248 bytes	1.003 ms	6.172 ms
Aceitação total	238 bytes	1.084 ms	6.542 ms
Recusa (5 serviços)	155 bytes	0.532 ms	2.952 ms
Aceitação (5 serviços)	150 bytes	0.416 ms	2.267 ms

Estes resultados demonstram que, tanto a criação do JWS, como a verificação das assinaturas digitais, apresentam tempos de execução reduzidos. No entanto, estes valores têm alguma variedade, mas podemos ver como a diferença do tamanho do *payload* provoca a diferença entre os tempos de criação do JWS e verificação da assinatura.

6.3 Discussão dos Resultados

A análise dos resultados permite constatar que o fluxo funcional de consentimento foi corretamente validado em vários cenários, desde a recusa total até à aceitação de todos os serviços, passando também por subconjuntos de consentimentos. Em todos os casos, a criação e validação de JWS foi executada com sucesso, confirmando a robustez da solução implementada.

No que respeita ao desempenho, verificou-se que o impacto do tamanho do *payload* nos tempos de

execução é significativa. Embora exista uma variação ligeira e proporcional entre os cenários de menor e maior dimensão, os valores registados mantêm-se sempre baixos, tanto para a criação de JWS como para a sua verificação no servidor.

Em termos práticos, estes resultados demonstram que a solução consegue cumprir os objetivos estabelecidos: assegurar a transparência e a auditabilidade dos consentimentos digitais através do registo JWS, preservando simultaneamente a integridade e a autenticidade das assinaturas, sem comprometer a experiência de utilização ou a escalabilidade do sistema.

Bibliografia

- Usercentrics A/S. Cookiebot cmp technical documentation, 2025. URL <https://support.cookiebot.com/hc/en-us/articles/360003774494-Why-does-your-scanner-say-that-I-have-more-than-50-pages>.
- Vitalik Buterin. A next-generation smart contract and decentralized application platform. *Ethereum White Paper*, 2014. URL <https://ethereum.org/en/whitepaper/>.
- CookieChimp. Cookiechimp api documentation and technical specifications, 2025. URL <https://cookiechimp.com/documentation/>.
- Cristòfol Daudén-Esmel, Jordi Castellà-Roca, and Alexandre Viejo. Blockchain-based access control system for efficient and gdpr-compliant personal data management. *Computer Communications*, 214, 2024.
- European Parliament and Council. Article 4 - definitions, 2016. URL <https://gdpr-info.eu/art-4-gdpr/>. General Data Protection Regulation (GDPR).
- Michael B. Jones, John Bradley, and Nat Sakimura. Json web signature (jws). RFC 7515, Internet Engineering Task Force (IETF), 2015. URL <https://datatracker.ietf.org/doc/html/rfc7515#section-7.2.1>.
- Klaro. Klaro! a simple consent manager, 2025. URL <https://github.com/kiprotect/klaro>. Acesso em: 07 set. 2025.
- Consent Manager. Technical implementation and architecture overview, 2025. URL <https://help.consentmanager.net/books/cmp/page/cross-device-consent-sharing>.
- Mozilla. Browser extensions, 2025. URL <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions>.
- Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *White Paper*, 2008. URL <https://bitcoin.org/bitcoin.pdf>.

Osano. Osano - data privacy platform, 2025. URL <https://docs.osano.com/hc/en-us/articles/22472101620756-Osano-Cookies>. Acesso em: 07 fev. 2025.

Cristiana Santos, Midas Nouwens, Michael Toth, Nataliia Bielova, and Vincent Roca. Consent management platforms under the gdpr: processors and/or controllers? In *APF 2021 - 9th Annual Privacy Forum*, Oslo, Norway, 2021. doi: 10.1007/978-3-030-76663-4_3. URL <https://inria.hal.science/hal-03169436v1>. HAL Id: hal-03169436.

S.R. Subramanya and B.K. Yi. Digital signatures. *IEEE Potentials*, 25(2):5–8, 2006. doi: 10.1109/MP.2006.1649003. URL https://www.researchgate.net/publication/3227862_Digital_signatures.

Tarteaucitron. Tarteaucitron - gdpr cookie consent manager, 2025. URL <https://tarteaucitron.io/en/>. Acesso em: 07 fev. 2025.

Nguyen Binh Truong, Kai Sun, Gyu Myoung Lee, and Yike Guo. Gdpr-compliant personal data management: A blockchain-based solution. *IEEE Transactions on Information Forensics and Security*, 15, 2020.

Vue. Working with webpack, 2025. URL <https://cli.vuejs.org/guide/webpack.html>.

Christian Wirth and Martin Kolain. Privacy by blockchain design: A blockchain-enabled gdpr-compliant approach for handling personal data. *Reports of the European Society for Socially Embedded Technologies (EUSSET)*, 2018.

Parte I

Apêndices

Apêndice A

Trabalho de apoio

Resultados auxiliares.

```
1 async function processConsent(consentData) {
2   console.log('Processing consent data with JWS:', consentData);
3
4   try {
5     // Step 1: Fetch server's certificate and pubKey
6     const serverCert = await fetch('http://127.0.0.1:3000/api/
      server_certificate');
7     const certPem = await serverCert.json();
8     const cert = forge.pki.certificateFromPem(certPem);
9     const publicKey = forge.pki.publicKeyToPem(cert.publicKey);
10
11    console.log('Received server public key data:', publicKey);
12
13    // Step 2: Import server's RSA public key
14    const serverPublicKey = await cryptoUtils.importRSAPublicKey(publicKey)
15    ;
16    console.log('Imported server RSA public key');
17
18    // Step 3: Load key pair for client
19    const clientSigningKeyPair = await cryptoUtils.loadSigningKeyPair();
20    const privKeyCrypto = await cryptoUtils.importPrivateKey(
21      clientSigningKeyPair.privKey);
22    console.log('Loaded client RSA signing keys');
23
24    // Step 4: Sign consentData
25    const clientSignature = await cryptoUtils.signData(privKeyCrypto,
26      consentData);
27
28    console.log('Client signed the consent');
29
30    const response = await fetch('http://127.0.0.1:3000/api/consent', {
31      method: 'POST',
32      headers: {
33        'Content-Type': 'application/json'
34      },
35      body: JSON.stringify({
36        signature: clientSignature,
37        consent: consentData,
38        pubkey: clientSigningKeyPair.pubKey
39      })
40    });
41
42    console.log('Client sent the info');
```

```

41     const result = await response.json();
42
43     console.log(result);
44
45     // Step 5: Verify server-signed JWS
46     if (result.success && result.serverSignedJWS) {
47         try {
48             const serverSignedPayload = await cryptoUtils.verifyServerJWS(
49                 result.serverSignedJWS,
50                 consentData,
51                 serverPublicKey
52             );
53
54             console.log('Server-signed JWS verified successfully');
55             console.log('Final payload:', serverSignedPayload);
56
57             return true;
58         } catch (verifyError) {
59             console.error('Server JWS verification failed:', verifyError);
60             return false;
61         }
62     } else {
63         console.error('Server error:', result.error);
64         return false;
65     }
66 } catch (error) {
67     console.error('Error processing consent:', error);
68     return false;
69 }
70 }

```

Apêndice B

Detalhes dos resultados

Detalhes de resultados cuja extensão comprometeria a legibilidade do texto principal.

Apêndice C

Listings

Se for o caso.

Apêndice D

Ferramentas

(Se for o caso)

Utilizadores de [L^AT_EX](#) devem consultar [TUG](#) , o grupo de utilizadores [T_EX](#) .

Coloque aqui informação sobre financiamento, projeto FCT, etc. em que o trabalho se enquadra. Deixe em branco caso contrário.