



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:



CC BY

<https://creativecommons.org/licenses/by/4.0/>

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 do consentimento de dados é uma das questões centrais para garantir a privacidade de todos os utilizadores, tendo em consideração regulamentações já existentes para o cumprimento da mesma. Esta dissertação explora as **Plataformas de Gestão de Consentimento (CMP)** existentes, destacando as suas características, limitações e potencialidades de forma a criar soluções mais auditáveis e transparentes. Assim, foi proposta uma solução que permite, não só recolher consentimentos de forma mais eficiente, mas também verificar as escolhas dos utilizadores. Inicialmente, foi realizada uma análise ao tratamento de dados, incluindo plataformas como Osano, Cookiebot, Klaro.js, entre outras, englobando soluções avançadas, como o caso do Google Consent Mode, e abordagens complementares para a gestão deste mesmo consentimento, baseadas em tecnologias como *blockchain* e contratos inteligentes. Contudo, foi possível verificar que as soluções já existentes, embora robustas, apresentam limitações no que toca à auditoria e transparência do fluxo de dados entre clientes e *provedores de serviço*. Os resultados obtidos permitem demonstrar uma prova de conceito de um sistema que integra a recolha de consentimento com mecanismos de auditoria, permitindo garantir que as autorizações recolhidas são armazenadas de forma verificável e imutável. Desta forma, esta dissertação propõe o desenvolvimento de uma solução que reforça a confiança dos utilizadores no tratamento dos seus dados, promovendo uma maior transparência e autonomia na gestão destes mesmos dados por parte do utilizador.

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. It was developed 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
2.5	Síntese do capítulo	15
3	Prova de Consentimento	17
3.1	Visão Geral da Arquitetura	17
3.2	Componentes	18
3.3	Modelo de confiança	19
3.3.1	Certificados digitais	19
3.3.2	Assinatura Digital	20
3.4	Registo do consentimento	20
3.5	Interacção Entre Componentes	21
3.6	Lógica de Funcionamento	22
3.7	Benefícios	22
3.8	Síntese do capítulo	23

4	Implementação da solução	24
4.1	Prova de Conceito	24
4.2	Protótipo e testes funcionais	25
4.2.1	Open-Source	25
4.2.2	JWS	27
4.2.3	Servidor e Website	29
4.2.4	Extensão no Cliente	31
4.2.5	Gestão de Certificados	31
4.2.6	Fluxo de Consentimento	33
4.2.7	Registo do consentimento	34
4.3	Síntese do capítulo	36
5	Conclusões e trabalho futuro	37
5.1	Conclusões	37
5.2	Trabalho futuro	38
5.2.1	Complementação do fluxo com <i>blockchain</i> e contratos inteligentes	38
5.2.2	Integração com keychain do sistema do cliente	39
5.2.3	Disponibilização da solução como projeto Open-source	39
6	Avaliação	40
6.1	Avaliação Funcional	40
6.2	Avaliação de Desempenho	43
6.3	Discussão dos Resultados	43
I	Apêndices	48
A	Trabalho de apoio	49
A.1	Implementação do Servidor	49
A.2	Processamento do Consentimento no Cliente	51
A.3	Criação de Certificados com OpenSSL	53
A.3.1	Criação da Root CA	53
A.3.2	Criação das Chaves e Certificados do Servidor e Cliente	53
A.4	Criação e Assinatura do JWS no Cliente	54

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 de prova de consentimento	22
5	Banner padrão do Klaro.js	33
6	Diagrama do protocolo	34
7	Extensão carregada no browser	41

Lista de Tabelas

1	Comparação das principais características das CMPs existentes	10
2	Resultados de desempenho em diferentes cenários de payload.	43

Acrónimos

API *Application Programming Interface.*

CA *Certificate Authority.*

CCPA *California Privacy Rights Act.*

CMP *Plataformas de Gestão de Consentimento.*

DC *Data Controller.*

DP *Data Processor.*

DR *Data Recipient.*

DS *Data Subject.*

GDPR *General Data Protection Regulation.*

RGPD *Regulamento Geral de Proteção de Dados.*

SP *Service Provider.*

UE *União Europeia.*

Capítulo 1

Introdução

Com a crescente utilização de plataformas online, a quantidade de dados pessoais recolhidos de utilizadores tem aumentado. Contudo, muitos utilizadores desconhecem como esses dados são recolhidos, processados e partilhados pelas entidades envolvidas com o mesmo processamento. 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)** (respetiva à União Europeia) e outras legislações de privacidade.

Plataformas de Gestão de Consentimento (**CMPs**) surgem como soluções para o processamento de dados, uma vez que dizem assegurar aos serviços das empresas que assim podem cumprir legislações de tratamento de dados tais como 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, no entanto, não conseguem fornecer uma visão clara sobre o que realmente acontece com os dados depois de o utilizador ter feito a escolha em relação ao seu consentimento, nem se a informação transmitida ao servidor está em conformidade com a escolha do utilizador. A falta de um método de auditoria transparente torna difícil o utilizador provar a não conformidade das empresas que não estão a respeitar o consentimento e a cumprir as regulamentações de privacidade.

1.1 Motivação

Uma das falhas dos modelos atuais reside no risco de que as escolhas de privacidade e as configurações de consentimento do utilizador não sejam efetivamente respeitadas no fluxo de dados. Num cenário de litígio ou disputa de privacidade, as ferramentas tradicionais não oferecem uma forma de prova criptográ-

fica, imutável e verificável. A motivação para este projeto surge então precisamente da necessidade de permitir a auditoria desse fluxo de dados, oferecendo uma forma de prova caso as escolhas do utilizador não forem efetivamente respeitadas. Uma das grandes limitações das ferramentas tradicionais é o facto de muitas delas serem soluções fechadas que não transmitem a maior transparência possível, dificultando a compreensão e auditoria do que acontece com os dados. Este projeto visa colmatar esta lacuna, fornece uma solução que não dependa da confiança cega nos registos internos do responsável pelo tratamento. Essa prova é robusta o suficiente para servir como evidência inquestionável caso seja detetado um incumprimento, permitindo que o utilizador verifique de forma independente o histórico completo e não adulterado do processamento dos seus dados.

1.2 Objetivos

Criar uma solução que permite não só recolher e gerir o consentimento dos utilizadores, mas também permitir auditar, em caso de não cumprimento dos consentimentos, os dados de forma transparente. Explorar as tecnologias de tratamentos de dados e permitir registar de forma imutável as interações e escolhas dos utilizadores, de modo a assegurar a conformidade e aumentar a confiança no processo. Desenhar e prototipar um sistema que ajuda na gestão dos consentimentos de um utilizador uma vez que se integra a recolha de consentimento com mecanismos de auditoria, utilizando estratégias criptográficas, de forma a criar registos verificáveis, imutáveis e não repudiáveis. O objetivo não é garantir que os **Service Provider (SP)** cumprirão integralmente regulamentos como o **RGPD**, pois a conformidade final depende das suas práticas internas. Em vez disso, a solução visa promover a transparência e, fundamentalmente, fornecer a prova técnica irrefutável (evidência) em caso que as decisões dos utilizadores foram (ou não foram) respeitadas no tratamento dos dados armazenados. Desta forma, o sistema capacita os utilizadores a provar a não-conformidade em caso de violação das escolhas de privacidade.

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 materializa 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 (*Firefox*), o desenvolvimento de um servidor, a integração com o **CMP** *open-source* Klaro.js e a gestão de certificados digitais, 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), bem como a motivação, os objectivos e o estudo do trabalho relacionado, apresentando algumas das soluções existentes para problemas semelhantes (Capítulo 1.4).

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 3.8), 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** (European Union, 2016). 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 encontra-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 existentes.

- **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.
- **Tarteaucitron.js & Klaro.js:** São soluções de código aberto que se destacam pela flexibilidade [Tarteaucitron \(2025\)](#). O Tarteaucitron.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 dos serviços existentes como também o registo explícito do consentimento do lado cliente via 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 um *provedor de serviços* mostrar que um website está em conformidade com os regulamentos pode ser representado no diagrama de atividades da Figura [2.1.1](#). 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.

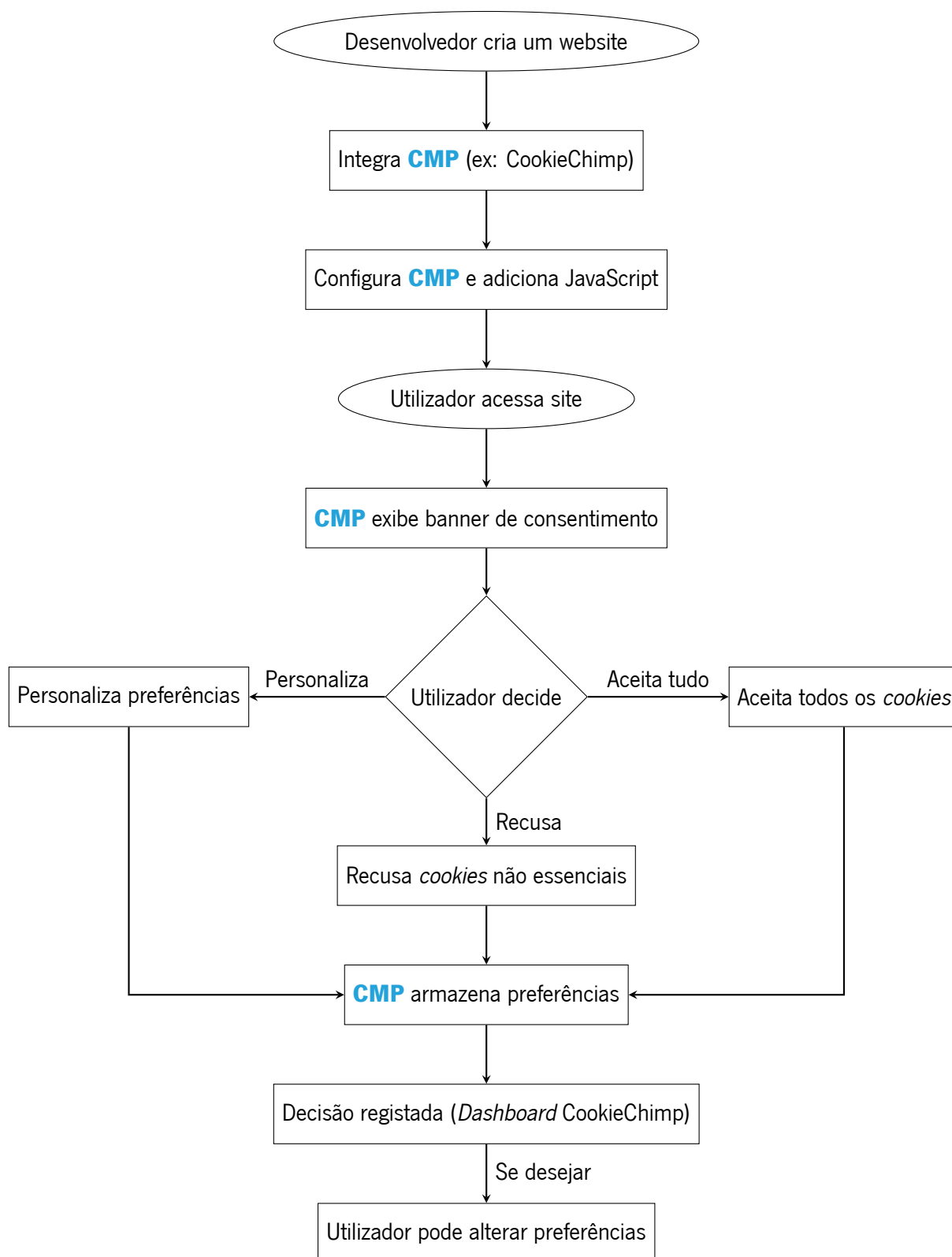



Figura 1: Fluxo de implementação de uma **CMP** num site.

O processo pode ser descrito da seguinte forma, detalhando os passos e as suas implicações:

1. Desenvolvedor cria um site e identifica a necessidade de conformidade: O primeiro passo no ciclo de vida do projeto web é a sua conceção e implementação. É nesta fase inicial que o desenvolvedor mapeia as cookies necessárias e tecnologias de rastreamento utilizados (próprios e de terceiros) para determinar o nível de risco e as obrigações regulamentares (ex: **RGPD**, **CCPA**).
2. Integração e Escolha da **CMP**: Para cumprir os requisitos legais de obtenção de consentimento, o desenvolvedor decide integrar uma **CMP**, como o CookieChimp.
3. Configuração da **CMP** e Adição do JavaScript: Após a escolha, o desenvolvedor configura a ferramenta para o seu domínio (definindo políticas, idioma, e aspeto do banner). O passo mais importante é a inserção de um pequeno código JavaScript fornecido pela **CMP** no cabeçalho (*header*) do site. Este script é o responsável por carregar o banner.
4. Acesso do Utilizador e Exibição do Banner: Quando um utilizador acessa o site, o script da **CMP** é o primeiro a ser executado, exibindo imediatamente um banner (figura 2.1.1) solicitando o consentimento.
5. Decisão do Utilizador: O utilizador tem controlo sobre a sua decisão, podendo:
 - Aceitar todos os cookies e tecnologias de rastreamento.
 - Recusar cookies não essenciais: A **CMP** exige cookies estritamente necessários para o funcionamento básico do site (como *tokens* de sessões) sejam carregados. Todos os de marketing, estatísticas ou personalização permanecem bloqueados.
 - Personalizar as suas preferências: O utilizador pode ligar/desligar categorias específicas de cookies (ex: marketing, estatísticas).
6. Armazenamento de Preferências e Bloqueio de Cookies: Independentemente da escolha, a **CMP** armazena a decisão numa cookie do lado do cliente com uma referência com a qual o servidor do CookieChimp interage de maneira a conseguir obter informação dos consentimentos guardados. Esta cookie serve como prova de consentimento e evita que o banner seja exibido novamente nas visitas subsequentes durante o período de validade do consentimento (ex: 6 ou 12 meses).
7. Registo da Decisão (Proof of Consent): A decisão detalhada do utilizador, a data e hora do consentimento (e até o ID de sessão para auditoria) são registadas e enviadas para o CookieChimp

e pode ser acedido através do dashboard do *provedor de serviço*, como pode ser visto na figura 2.1.1.


8. Alteração de Preferências: Para cumprir o princípio de que o consentimento deve ser tão fácil de retirar quanto de dar, o utilizador pode alterar as suas preferências a qualquer momento. Isto é geralmente feito através de um botão ou link discreto, mas acessível (ex: um ícone flutuante ou link no rodapé), que reabre o banner do **CMP**.

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				
IP Region	Porto				
User ID	---				
Accept Type	<input checked="" type="radio"/> All				
GPC Preference	---				
Accepted Categories	essential marketing				
Rejected Categories	---				
Accepted At	2025-01-20 21:48:23 UTC				

WordPress			
Name	Company	Description	Duration
comment_author_*	WordPress	Used to store the user's name and email...	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

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

Desta forma, as **CMP**s transmitem transparência e conformidade com as regulamentações de privacidade aos *provedores de serviço* e permitir que os utilizadores tenham escolhas sobre quais dos seus dados pessoais quer partilhar.

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 capacidade de uma auditoria mais rigorosa do processo, garantindo que os utilizadores possam comprovar a integridade

da plataforma. Além disso, a auditabilidade é uma qualidade essencial, uma vez que possibilita aos os utilizadores a possibilidade de comprovar se os seus pedidos são efetivamente respeitados, fornecendo uma camada adicional de confiança ao assegurar que ele tem como provar de como as suas informações foram guardadas sem possível repúdio.

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 mais complexos.

A tabela 2.2 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.

No domínio da gestão do consentimento, esta é também uma lacuna presente em praticamente todas as **CMPs**. A maioria das plataformas oferece funcionalidades e estatísticas orientadas para o *Provedor de Serviços*, mas do lado do utilizador não existe uma ferramenta que permita uma gestão eficiente dos consentimentos. Esta limitação impede o armazenamento e a consulta sistemática dos consentimentos

estabelecidos entre ambas as entidades, comprometendo a transparência e o controlo efetivo por parte do utilizador.

Com isto, outra limitação crítica é a ausência de mecanismos robustos de confiança. 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 **CMPs** 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 o instrumento legislativo da **União Europeia (UE)** que define as regras relativas à proteção das pessoas singulares no que respeita ao tratamento de dados pessoais e à livre circulação desses dados. Reconhecendo que a proteção de dados pessoais constitui um direito fundamental, o **RGPD** visa harmonizar as legislações dos Estados-Membros, garantindo a privacidade dos cidadãos e permitindo, dentro desse quadro, a circulação segura e legal de dados no mercado interno da União Europeia.

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 de-

termina 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 obrigatória 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 (Ribeiro et al. (2025), Ramos and Silva (2019)).

Por fim, uma das características mais relevantes do **RGPD** é o seu enfoque na responsabilização e transparência no tratamento de dados pessoais. O regulamento exige que os **Service Provider (SP)** sejam capazes de demonstrar a conformidade com as disposições legais e de documentar os acordos estabelecidos com os *titulares dos dados*. Este princípio de responsabilização reforça a importância de adoptar abordagens inovadoras e ferramentas tecnológicas que promovam maior confiança, rastreabilidade e conformidade com as normas regulamentares.

2.4 Blockchain e Contratos Inteligentes

Uma abordagem promissora para a auditoria de consentimentos de dados é a utilização de *blockchain* e contratos inteligentes. A *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 a verificação e aplicação das condições definidas pelos utilizadores, garantindo que as regras definidas são cumpridas de forma autónoma e transparente.

A tecnologia *blockchain* distingue-se por características como descentralização, transparência, integridade e imutabilidade, tornando o registo de dados resistente a alterações não autorizadas e à manipulação por qualquer entidade central. 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 (Frank, 2018).

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.

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

- **Transparência e Imutabilidade:** Em blockchains públicas, o registo de consentimentos é transparente, permitindo que qualquer participante verifique as transações, enquanto a estrutura imutável garante que os dados não possam ser 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.

No entanto, apresenta também algumas desvantagens. Um dos principais desafios associados à utilização da *blockchain* na gestão de consentimento decorre precisamente da sua transparência. Os desenvolvedores têm de considerar cuidadosamente os inconvenientes de tornar toda a informação acessível publicamente e encontrar soluções que permitam proteger os dados sensíveis. Por exemplo, dependendo da forma como este registo é armazenado, é possível rastrear os serviços que um utilizador visita, permitindo inferir os seus interesses pessoais. Para mitigar este risco, seria necessária a inclusão de uma camada criptográfica adicional, de modo a preservar o anonimato e assegurar que os dados sensíveis permanecem privados, sem comprometer a transparência e a integridade do sistema. Outro desafio associado em aplicações como a gestão de consentimento, prende-se com a eficiência e os custos. Cada transação não é processada por uma única unidade, mas por toda a rede, o que implica uma repetição massiva das operações e um custo significativamente superior ao de sistemas tradicionais. Para além disso, a velocidade da rede é limitada pelos mecanismos de consenso, que tendem a criar um trade-off entre descentralização e rapidez. Adicionalmente, a introdução de aplicações baseadas em *blockchain* enfrenta barreiras regulatórias e legais, nomeadamente quanto à validade e enquadramento dos *smart contracts*. Estes fatores devem ser ponderados cuidadosamente, de modo a justificar o valor acrescentado da tecnologia face às suas limitações operacionais e jurídicas.

Contudo, 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.

2.5 Síntese do capítulo

Neste capítulo são 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 é apresentada a solução 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

Prova de Consentimento

O presente capítulo descreve a arquitectura conceptual da solução proposta para a gestão e prova de consentimento digital. Pretende-se apresentar a visão geral do sistema, destacando os principais componentes, a lógica de funcionamento e os mecanismos criptográficos que garantem propriedades fundamentais como autenticidade, integridade, não repúdio, transparência e auditabilidade. Através desta abordagem, é possível assegurar que cada decisão do utilizador é registada de forma verificável e imutável.

3.1 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 juntamente com o *Provedor de Serviços*, assegurando que nenhuma das partes pode manipular ou negar a informação posteriormente. Para tal, existem duas entidades principais:

- **Data Subject (DS)**: o utilizador final que interage com a interface web do serviço e fornece o consentimento no qual tem uma extensão de navegador à escuta dos eventos do 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 um servidor para a troca de informação. Isto é, 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:

1. O **DS** interage com o banner de consentimento apresentado pelo **CMP** escolhido 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*

Este processo pode ser observado de forma mais clara na Figura 3.5, que ilustra o fluxo completo de interações entre as diferentes entidades do sistema.

Desta forma, o sistema garante quatro propriedades fundamentais. Em primeiro lugar, assegura a *transparência*, uma vez que todos os passos do processo podem ser verificados tanto pelo utilizador como pelo prestador de serviços. Em segundo lugar, garante a *autenticidade* e *integridade* dos consentimentos, através do uso de assinaturas digitais que impedem alterações e confirmam a proveniência das entidades envolvidas. A terceira propriedade é o *não repúdio*, que impede o **DS** de negar a sua decisão e o **SP** de alegar que não recebeu ou validou o consentimento. Por fim, o sistema promove a *auditabilidade*, mantendo um histórico de consentimentos acessível a ambas as partes, o que permite cumprir os requisitos legais e regulatórios em matéria de proteção de dados.

3.2 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 feitos pelo utilizador e trata, no background, do processo de assinatura digital como também a autenticação do utilizador através do envio do certificado digital do mesmo. Por fim, existe um servidor com o qual são trocados os certificados e assinaturas, permitindo que, no final do processo, seja obtido um registo estruturado de consentimento auditável e verificável. Este é responsável por validar as assinaturas recebidas e criar um registo imutável com o consentimento assinado por ambas as partes.

3.3 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**.

3.3.1 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 intermédias ou diretamente de clientes e servidores, porém esta última trata-se de uma má prática, normas internacionais como a ETSI EN 319 411 (**ETS**, 2023) proíbem explicitamente o uso direto da Root CA para emitir certificados operacionais. Este mecanismo hierárquico garante que, ao receber um certificado, é possível verificar a sua autenticidade através da cadeia de confiança estabelecida pela autoridade certificadora.

3.3.2 Assinatura Digital

A assinatura digital é um mecanismo criptográfico baseado em criptografia de chave pública, 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.

3.4 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 sobre 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;

A ausência de cifragem no registo de consentimento simplifica o processo de verificação e auditoria, mas aumenta o risco de exposição de informação pessoal. Mesmo que o registo não contenha dados diretamente identificáveis, como o nome ou o e-mail, pode incluir identificadores indiretos, por exemplo, identificadores únicos, que podem permitir reconhecer ou reidentificar o utilizador. Para reduzir este risco e garantir conformidade com o princípio da confidencialidade previsto nas legislações, recomenda-se a aplicação de técnicas complementares de pseudonimização ou cifragem seletiva, de modo a equilibrar a transparência com a proteção da privacidade do utilizador.

3.5 Interacção Entre Componentes

A figura 3.5 ilustra o fluxo completo de troca de mensagens e assinaturas durante o processo de consentimento. O sistema envolve três componentes principais: o *cliente*, que inclui a extensão do navegador para capturar e assinar digitalmente os consentimentos; a *interface web* do *prestador de serviços*, que apresenta o *banner* de consentimento; e o *servidor*, responsável por validar e assinar os registos de consentimento.

Quando o utilizador acede ao website, a interface web apresenta o *banner* de consentimento e juntamente envia o certificado do servidor. O utilizador preenche o banner e, através da extensão do navegador, assina digitalmente o consentimento utilizando a sua chave privada. A extensão prepara então o objeto final do consentimento, que inclui a assinatura do utilizador, e envia-o para o servidor juntamente com o certificado do cliente.

O servidor valida a assinatura do utilizador e o respetivo certificado, gera a sua própria assinatura digital sobre o consentimento e cria um registo final que agrega ambas as assinaturas. Este registo é devolvido ao cliente, que valida a assinatura do servidor e mantém uma cópia do consentimento de forma verificável.

Desta forma, o fluxo garante que, tanto o cliente como o servidor, dispõem de provas verificáveis e mutuamente validadas do consentimento, assegurando transparência, integridade e auditabilidade ao longo de todo o processo.

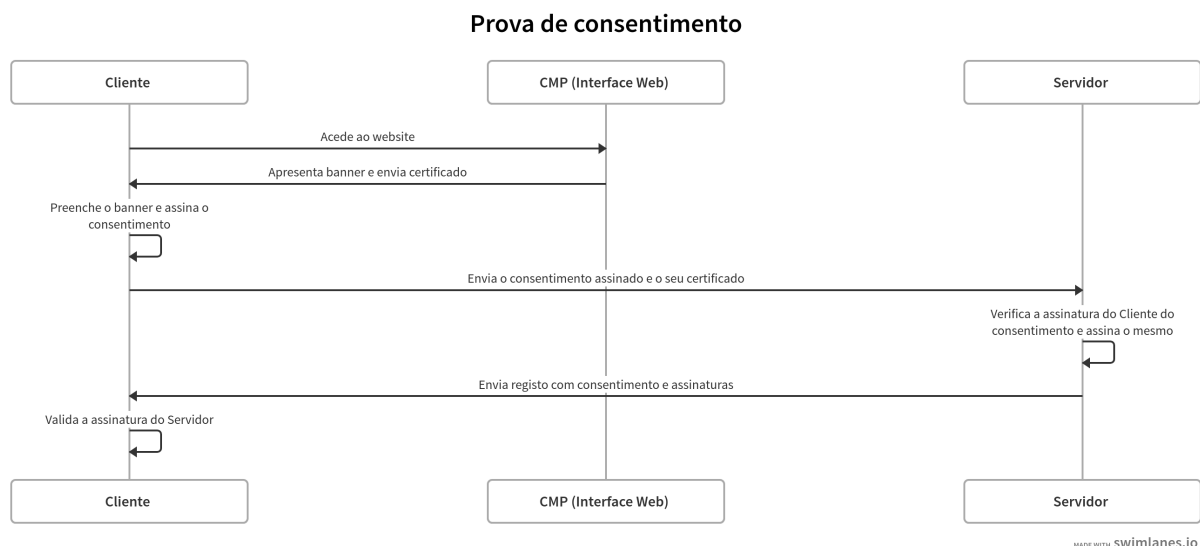


Figura 4: Diagrama do protocolo de prova de consentimento

3.6 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 *audibilidade*, uma vez que o histórico completo permanece acessível tanto localmente, junto de cada entidade participante, como opcionalmente através de um serviço de terceiros responsável pela verificação ou conservação dos registos.

3.7 Benefícios

A *Prova de Consentimento* proposta apresenta benefícios distintos para os diferentes intervenientes. Do ponto de vista do utilizador, este 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.

3.8 Síntese do capítulo

Neste capítulo foi apresentada a arquitectura conceptual da solução de prova de consentimento, detalhando os principais componentes, a lógica de funcionamento e os mecanismos criptográficos, como certificados digitais e assinaturas digitais. Através da Prova de Conceito, demonstrou-se a viabilidade do modelo conceptual, validando a autenticidade, integridade, não repúdio, transparência e auditabilidade dos registos de consentimento num ambiente simplificado, sem recorrer ainda a uma implementação completa.

O capítulo seguinte descreve a implementação prática desta arquitectura, materializada num protótipo funcional que integra os princípios validados na Prova de Conceito. Nesta fase, são exploradas tecnologias reais e fluxos completos, permitindo avaliar o desempenho, a interoperabilidade e a experiência de utilização, evidenciando os benefícios e limitações do sistema face às limitações identificadas no trabalho relacionado ([1.4](#)).

Capítulo 4

Implementação da solução

Neste capítulo apresenta-se a prova de conceito da solução proposta, demonstrando a viabilidade da arquitetura definida no capítulo anterior, seguida da implementação de um protótipo funcional. O protótipo permite testar os principais componentes do sistema, as tecnologias utilizadas e o fluxo de comunicação que assegura a recolha, assinatura, validação e registo do consentimento digital. Por fim, são apresentados os resultados dos testes funcionais, que permitem avaliar se a implementação cumpre os objetivos definidos para o sistema.

4.1 Prova de Conceito

O objetivo é demonstrar a viabilidade técnica e conceptual da *Prova de consentimento* apresentada no Capítulo 3, validando os princípios de autenticidade, integridade, não repúdio, transparência e auditabilidade sem recorrer a uma implementação completa.

Para efeitos da prova de conceito, o sistema é reduzido aos componentes essenciais: o cliente, que representa o utilizador final (**DS**), o servidor, que representa o prestador de serviços (**SP**), e o fluxo de comunicação entre ambos. O cliente simulado é responsável por gerar o registo de consentimento e assinar digitalmente o conteúdo, criando uma prova verificável da decisão do utilizador. O servidor simulado recebe esse registo, valida a assinatura do cliente, acrescenta a sua própria assinatura e mantém um registo final verificável. O fluxo de comunicação é simplificado, permitindo validar a consistência do registo e a integridade das assinaturas sem necessidade de implementação completa de uma extensão de navegador ou interface web funcional.

O fluxo conceptual inicia-se com a geração do registo de consentimento pelo cliente, contendo as decisões relativas aos serviços. O registo é então assinado digitalmente com a chave privada do cliente e enviado ao servidor, que valida a assinatura, garantindo integridade e autenticidade. Em seguida, o servidor adiciona a sua própria assinatura digital, reforçando a prova e assegurando o não repúdio. O

registo final, agora assinado por ambas as partes, é armazenado ou devolvido ao cliente, que pode verificar novamente as assinaturas. Qualquer alteração posterior no registo invalida as assinaturas, garantindo verificabilidade e auditabilidade do processo.

Na prova de conceito, o registo de consentimento é representado como um objeto estruturado, contendo o *payload* com as decisões do utilizador, a assinatura digital do cliente e a assinatura digital do servidor. Esta abordagem demonstra que o sistema suporta múltiplas assinaturas sobre o mesmo conteúdo, conforme previsto na *JSON Serialization* do JWS, assegurando verificabilidade e auditabilidade mesmo num modelo simplificado.

A prova de conceito permite confirmar que é possível gerar, assinar e verificar registos de consentimento de forma independente, validando a integridade e autenticidade dos dados e assegurando o não repúdio. Além disso, evidencia que a arquitetura conceptual é flexível e pode ser posteriormente implementada num protótipo funcional com componentes reais, incluindo extensões de navegador, servidores e CMPs diferentes. Assim, esta demonstração serve como validação experimental da arquitetura, reforçando a confiança na abordagem e evidenciando os benefícios do modelo proposto.

4.2 Protótipo e testes funcionais

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.

4.2.1 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.

Para alcançar este objetivo sem a necessidade da criação de um **CMP**, a solução adota uma abordagem que combina transparência, auditabilidade e eficiência com o uso de um **CMP** Open-source. 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.

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 consentimentos são tratados. Por outro lado, **CMPs** Open-source oferecem a vantagem de serem transparentes, acessíveis e adaptáveis, permitindo que o código-fonte seja inspecionado, auditado e modificado para se adequar a diferentes contextos.

Plataformas Open-source como o **Klaro.js** oferecem uma maior flexibilidade: permitem que a solução seja utilizada como base para o desenvolvimento de sistemas próprios, sem a necessidade de criar um CMP do zero. Isto reduz o esforço de implementação e facilita a integração com diferentes navegadores, serviços ou fluxos de consentimento. Para o utilizador, o benefício reside não apenas na possibilidade de verificar e auditar o funcionamento da ferramenta, mas também na garantia de que os seus consentimentos são processados de forma consistente e que a implementação pode ser adaptada para reforçar a sua privacidade e controlo sobre os dados.

4.2.2 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](#)).

Descrição Geral

De acordo com o RFC 7515, um JWS representa conteúdos protegidos com assinaturas digitais ou códigos de autenticação de mensagem (MAC) ([of Standards and Technology](#)), 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 ([Auth0, 2025](#))), o tipo de objeto (por exemplo, JWT), e possivelmente outros parâmetros relevantes.
2. *Payload*: o conteúdo a assinar que é codificado em Base64URL.
3. *Signature*: o resultado da assinatura digital aplicada ao *header* e *payload*, garantindo integridade e autenticidade.

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, tendo sido escolhido devido às suas vantagens de segurança face ao RS256. O PS256 utiliza o esquema de preenchimento probabilístico (PSS), gerando assinaturas diferentes para o mesmo *header* e *payload*, o que aumenta a resistência contra ataques a assinaturas determinísticas e fornece maior robustez criptográfica (Brady, 2020).
- 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.
- A presença de múltiplas assinaturas segue o modelo da *JSON Serialization* do JWS, que suporta várias assinaturas sobre o mesmo *payload*.

Vantagens neste contexto

Neste contexto, a utilização do JWS oferece várias vantagens. Em primeiro lugar, garante a integridade e autenticidade do consentimento, uma vez que qualquer modificação no *payload* ou nas assinaturas invalida o JWS, protegendo contra manipulação. Em segundo lugar, assegura o não repúdio, uma vez que as assinaturas do cliente e do servidor permitem verificar que o consentimento foi efectivamente fornecido pelo utilizador e validado pelo servidor. Além disso, tanto o cliente como o servidor conseguem verificar de forma independente a autenticidade do consentimento, reforçando a confiança no processo. Por fim, o JWS segue padrões abertos e é interoperável, o que facilita a integração com outras ferramentas e sistemas, permitindo que os registos de consentimento sejam utilizados de forma consistente em diferentes contextos e aplicações.

- **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.

4.2.3 Servidor e Website

Para capturar a interação do **DS**, é disponibilizado um website estático em HTML, no qual se integra o *script* do KlaroJS, seguindo a prática comum na implementação de **CMPs**. É possível 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 relevância quais os serviços a ser utilizados, foram mantidos os valores defaults.

Pode-se ver um exemplo de serviço configurado (Listagem 4.1):

```
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   },
```

Listing 4.1: Exemplo de serviço configurado no KlaroJS

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 configuração. Sendo assim só foi necessário chamar estes dois *scripts* para implementar o serviço (ver Listagem 4.2).

```
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>
```

Listing 4.2: Integração do KlaroJS com a configuração local

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 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.

A implementação completa dos *endpoints* e da lógica de verificação e assinatura pode ser consultada no Apêndice [A.1](#).

Deste modo, o servidor é 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.

4.2.4 Extensão no Cliente

Do lado do cliente desenvolveu-se uma extensão para o navegador *Firefox*, responsável pela gestão da troca de consentimentos. Esta foi implementada em JavaScript ([Mozilla, 2025](#)). Inicialmente foi utilizado o `manifest v2`. O `manifest` é um ficheiro de configuração obrigatório em qualquer extensão de navegador, que define informações essenciais sobre a extensão, como nome, versão, permissões, scripts a executar e outros recursos necessários para o seu funcionamento. O `manifest v3` foi uma atualização do formato usado pelas extensões, focada sobretudo em reforçar a segurança e privacidade, limitando certas funcionalidades usadas por ad blockers e scripts de terceiros. No entanto, para a extensão desenvolvida a migração do v2 para o v3 não trouxe diferenças práticas significativas, uma vez que o seu funcionamento não depende das funcionalidades que foram restringidas nesta atualização.

Module Bundlers

Durante o desenvolvimento da extensão em JavaScript, foi necessário recorrer a um *module bundler* para gerir as bibliotecas externas e garantir compatibilidade entre navegadores. Para tal, utilizou-se o Vite, ferramenta padrão do Vue.js, que simplifica a integração das dependências e a preparação do código para testes e distribuição. O Vite permitiu organizar o código e as dependências de forma eficiente, facilitando o desenvolvimento e a disponibilização da extensão.

4.2.5 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. Os certificados utilizados na solução são do tipo **X.509 v3**. Com isto, é necessário também que cada uma das entidades tenha a sua chave privada.

Detalhes sobre a criação dos certificados e das chaves podem ser encontrados no Apêndice [A.3](#).

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 e carregados pela própria extensão, como é visível na Listagem [4.3](#).

```
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 ...

```

Listing 4.3: Carregamento do certificado e da chave privada do cliente a partir do *local storage*

A integração com o módulo `node-forge` foi utilizada para o tratamento e gestão de chaves criptográficas e certificados digitais, suportando as operações de assinatura e verificação do sistema. 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 excerto de código (Listagem 4.4) 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 }

```

Listing 4.4: Carregamento das chaves RSA do servidor e do certificado associado

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.

4.2.6 Fluxo de Consentimento

O processo inicia-se quando o utilizador interage com o *banner* de consentimento (aceitação ou rejeição), disponibilizado pelo *Klaro.js* (Figura 4.2.6).

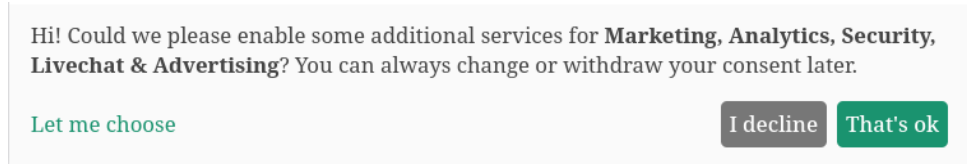


Figura 5: Banner padrão do Klaro.js

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 digital 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 registar localmente o consentimento.

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

Para maior detalhe sobre a implementação prática deste fluxo, incluindo a captura do evento de consentimento, assinatura digital, envio ao servidor e verificação do JWS, consulte o Apêndice A.2, onde a função `processConsent` é apresentada na íntegra.

O fluxo descrito pode ser visualizado no diagrama da Figura 4.2.6.

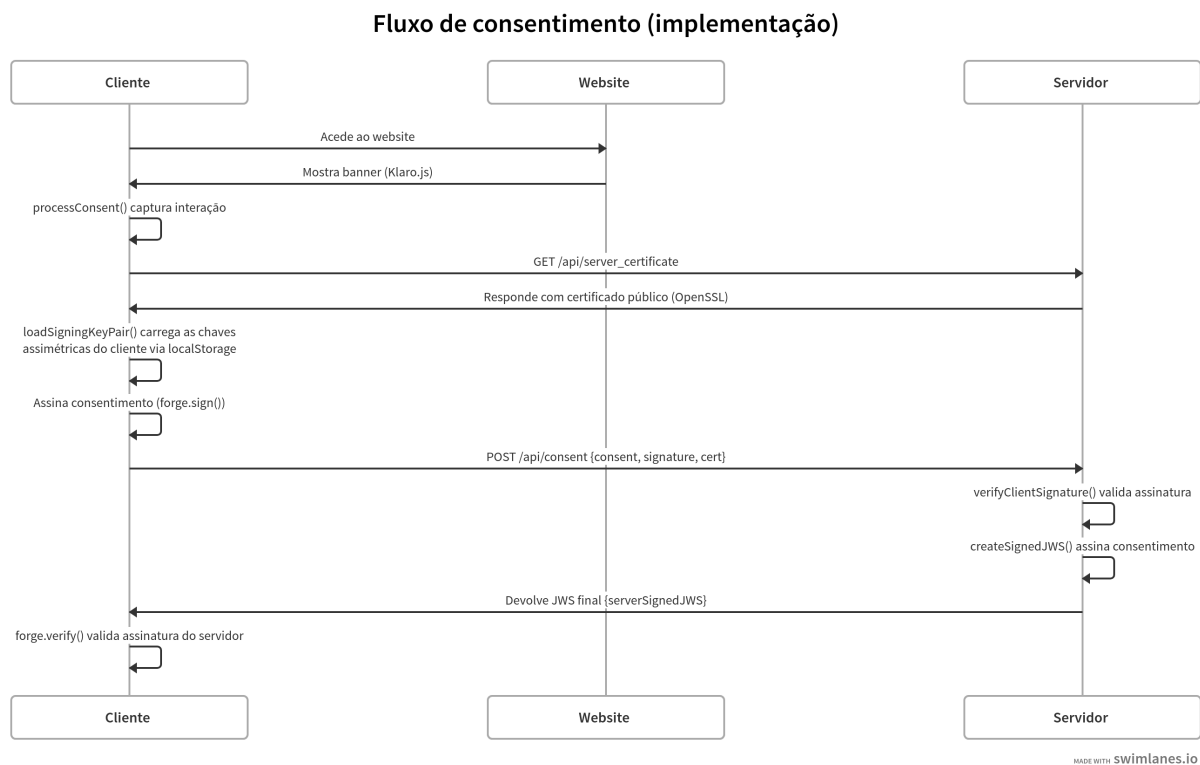


Figura 6: Diagrama do protocolo

4.2.7 Registo do consentimento

O registo do consentimento, juntamente com as respetivas assinaturas, constitui o resultado final do fluxo implementado. Trata-se de um registo digital que comprova a aceitação ou rejeição, pelo utilizador, de determinados serviços ou finalidades. Este registo é implementado sob a forma de um *JSON Web Signature (JWS)*, que encapsula o *payload* com os detalhes do consentimento do utilizador, a assinatura digital do cliente — garantindo que foi efetivamente o utilizador a autorizar — e a assinatura digital do servidor, que confirma a validação e integridade do consentimento.

O JWS é mantido por ambas as entidades, cliente e servidor, permitindo consultas futuras, auditoria e eventual revogação do consentimento. Este mecanismo assegura a *autenticidade*, uma vez que a assinatura do cliente comprova que o consentimento foi emitido pela pessoa correta; a *integridade*, já que qualquer alteração no *payload* invalida as assinaturas e impede manipulação; e o *não repúdio*, pois o cliente não pode negar a sua decisão devido à assinatura digital inequívoca.

Podemos consultar no apêndice [A.4](#) a implementação do JWS do lado do cliente.

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": "eyJjb25zZW50cyI6eyJ0d210dGVyIjpmYWxzZSw ...",
3   "signatures": [
4     {
5       "header": {"typ": "JWT", "alg": "PS256"},
6       "signature": "b1Xn5AaxYZWNfHoeL-SWTAySbT8yFWjJiPTK_rlIoPwTdukp9wpm
          ... "
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. Deste modo, o mecanismo implementado cumpre os objetivos definidos na criação desta prova de conceito, ao fornecer uma forma de registar e gerir consentimentos de forma transparente e técnica, garantindo a criação de registos verificáveis, imutáveis e não repudiáveis.

A solução permite não só a recolha e gestão dos consentimentos dos utilizadores, mas também possibilita auditar os dados em caso de não cumprimento, promovendo a transparência e fornecendo evidência técnica irrefutável de que as decisões dos utilizadores foram, ou não foram, respeitadas no tratamento dos dados. Embora não garanta a conformidade integral dos prestadores de serviços com regulamentos como o **RGPD**, o sistema permite aos utilizadores demonstrar a não-conformidade em caso de violação das suas escolhas de privacidade, cumprindo assim os objetivos principais do projeto. Os testes funcionais e de desempenho descritos no Capítulo 6 confirmam a viabilidade do modelo, validando o fluxo de consentimento, a criação e verificação de JWS e a robustez das assinaturas digitais.

4.3 Síntese do capítulo

Neste capítulo foram apresentadas duas fases complementares da implementação da solução proposta. A prova de conceito permitiu validar a arquitetura conceptual, demonstrando a viabilidade técnica dos princípios de integridade, autenticidade, não repúdio, transparência e auditabilidade num modelo simplificado. Seguiu-se o desenvolvimento do protótipo funcional, integrando cliente, servidor e interface web, com a utilização de certificados digitais, assinaturas e o formato JWS para criar registos de consentimento auditáveis e verificáveis. Foram realizados testes funcionais e de desempenho, descritos no Capítulo 6, que confirmaram o correto funcionamento do fluxo de consentimento e a robustez das assinaturas digitais.

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 e verificável caso fosse necessário. 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 sugere funcionalidades 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 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.

Importa salientar que, na integração com *blockchain*, apenas identificadores ou *hashes* dos consentimentos seriam registados, enquanto os conteúdos permaneceriam cifrados. Caso os dados completos fossem colocados diretamente na blockchain sem cuidado, existiria o risco de rastreabilidade das ações dos utilizadores, permitindo a monitorização das suas escolhas de consentimento e atividade online. Ao manter os conteúdos cifrados, esta abordagem assegura que a imutabilidade e auditabilidade do registo não comprometem a privacidade nem a confidencialidade dos utilizadores.

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

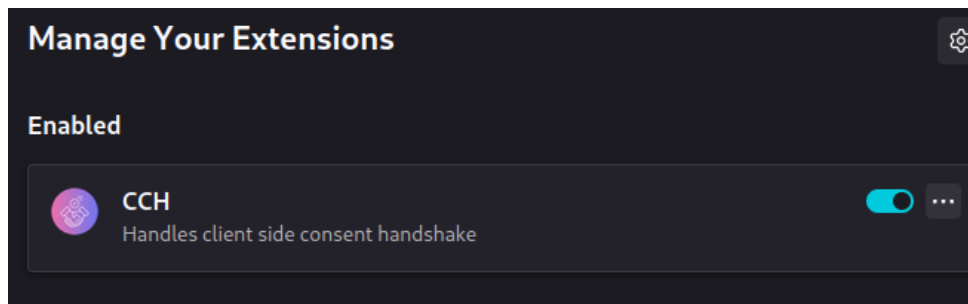


Figura 7: Extensão carregada no browser

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 significativo. 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

Electronic signatures and infrastructures (esi); policy and security requirements for trust service providers issuing certificates; part 1: General requirements, October 2023. URL https://www.etsi.org/deliver/etsi_en/319400_319499/31941101/01.04.01_60/en_31941101v010401p.pdf. Part 1: General requirements.

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>.

Auth0. Signing algorithms, 2025. URL <https://auth0.com/docs/get-started/applications/signing-algorithms>. Acedido em: 19 de Outubro de 2025.

Scott Brady. Jwts: Which signing algorithm should i use? <https://www.scottbrady.io/jose/jwts-which-signing-algorithm-should-i-use>, 2020. Acedido em: 19 de Outubro de 2025.

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).

European Union. 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),

2016. URL <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679>. Official Journal of the European Union, L 119/1.
- Fabian Frank. Consent management on the ethereum blockchain. Thesis, University of Twente, Enschede, October 2018. URL <https://purl.utwente.nl/essays/76745>.
- 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>.
- National Institute of Standards and Technology. Glossary: Message authentication code (mac). https://csrc.nist.gov/glossary/term/message_authentication_code. Acedido em: 19 de Outubro de 2025.
- Osano. Osano - data privacy platform, 2025. URL <https://docs.osano.com/hc/en-us/articles/22472101620756-Osano-Cookies>. Acesso em: 07 fev. 2025.
- Luis Felipe M Ramos and João Marco C Silva. Privacy and data protection concerns regarding the use of blockchains in smart cities. In *Proceedings of the 12th international conference on theory and practice of electronic governance*, pages 342–347, 2019.
- Diogo Ribeiro, Vitor Fonte, Luis Felipe Ramos, and João Marco Silva. Assessing the information security posture of online public services worldwide: Technical insights, trends, and policy implications. *Government Information Quarterly*, 42(2):102031, 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

Este capítulo apresenta o código e componentes desenvolvidos que suportam o funcionamento do sistema, incluindo implementações no lado do servidor e do cliente.

A.1 Implementação do Servidor

Este apêndice apresenta a implementação do *back-end* responsável pela gestão do processo de consentimento. O servidor, desenvolvido em *Node.js* com o *framework Express*, disponibiliza dois *endpoints* principais: um para a distribuição do certificado e outro para o processamento do consentimento assinado pelo cliente.

O código seguinte demonstra a lógica central associada a estes *endpoints*, incluindo:

- Disponibilização do certificado do servidor para validação de comunicações seguras;
- Receção, verificação e assinatura do consentimento submetido pelo cliente;
- Geração do registo final no formato *JSON Web Signature* (JWS), que comprova a integridade e autenticidade das trocas.

O fragmento de código apresentado foi simplificado para fins de legibilidade, mantendo apenas as operações essenciais do processo.

```
1 this.app.get('/api/server_certificate', (_, res) => {
2   try {
3     const certificate = this.serverCert;
4     res.json(certificate);
5     console.log('Server certificate sent to client');
6   } catch (error) {
7     console.error('Error sending certificate', error);
8     res.status(500).json({ error: 'Failed to get certificate' });
9   }
10 });
11
12 // Process consent JWS from client
13 this.app.post('/api/consent', (req, res) => {
14   try {
```

```

15     console.log('Processing consent JWS...');
16     const result = this.processClient(req.body);
17
18     res.json({
19         success: true,
20         serverSignedJWS: result.jws,
21         message: 'JWS processed and server-signed successfully'
22     });
23
24     console.log('JWS processed and server-signed');
25 } catch (error) {
26     console.error('Error processing JWS:', error);
27     res.status(400).json({
28         success: false,
29         error: error.message
30     });
31 }
32 });
33
34 processClient(clientInfo) {
35     console.log('Received client info...');
36
37     // Step 1: Verify client signature
38     this.verifyClientSignature(clientInfo.pubkey, clientInfo.consent,
39         clientInfo.signature)
39
40     // Step 2: Create server-signed JWS
41     const signedJWS = this.createSignedJWS(clientInfo);
42     console.log('JWS created');
43
44     return { jws: signedJWS };
45 }

```

A.2 Processamento do Consentimento no Cliente

Implementação da função `processConsent`, responsável pela gestão do fluxo de consentimento do lado do cliente, incluindo a captura do evento do banner, assinatura digital do consentimento, comunicação com o servidor e verificação do JWS devolvido. Este código ilustra de forma prática como o processo descrito na Secção 4.2.6 é concretizado na extensão do navegador.

```
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/
7       server_certificate');
8     const certPem = await serverCert.json();
9     const cert = forge.pki.certificateFromPem(certPem);
10    const publicKey = forge.pki.publicKeyToPem(cert.publicKey);
11
12    console.log('Received server public key data:', publicKey);
13
14    // Step 2: Import server's RSA public key
15    const serverPublicKey = await cryptoUtils.importRSAPublicKey(publicKey)
16    ;
17    console.log('Imported server RSA public key');
18
19    // Step 3: Load key pair for client
20    const clientSigningKeyPair = await cryptoUtils.loadSigningKeyPair();
21    const privKeyCrypto = await cryptoUtils.importPrivateKey(
22      clientSigningKeyPair.privKey);
23    console.log('Loaded client RSA signing keys');
24
25    // Step 4: Sign consentData
26    const clientSignature = await cryptoUtils.signData(privKeyCrypto,
27      consentData);
28
29    console.log('Client signed the consent');
30
31    const response = await fetch('http://127.0.0.1:3000/api/consent', {
32      method: 'POST',
33      headers: {
34        'Content-Type': 'application/json'
35      },
36      body: JSON.stringify({
37        signature: clientSignature,
38        consent: consentData,
39        pubkey: clientSigningKeyPair.pubKey
40      })
41    });
42
43    console.log('Client sent the info');
44
45    const result = await response.json();
46
47    console.log(result);
48  }
49 }
```



```

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 }

```

A.3 Criação de Certificados com OpenSSL

Para efeitos de teste, foram gerados certificados e chaves privadas utilizando o OpenSSL. A criação seguiu os seguintes passos:

A.3.1 Criação da Root CA

```
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
```

A.3.2 Criação das Chaves e Certificados do Servidor e Cliente

```
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
4 -CAcreateserial -out {server/client}.crt -days 825 -sha256
```

Estes certificados foram posteriormente utilizados pela aplicação para autenticação mútua entre servidor e cliente, garantindo a segurança das comunicações.

A.4 Criação e Assinatura do JWS no Cliente

Implementação responsável pela criação e assinatura do *JSON Web Signature* (JWS) no lado do cliente. O código demonstra como o consentimento do utilizador é encapsulado num objeto estruturado, assinado digitalmente com a chave privada do cliente e posteriormente validado com a chave pública do servidor. A implementação tem como objetivo garantir as propriedades de autenticidade, integridade e não repúdio do registo de consentimento, servindo como evidência verificável da decisão do utilizador no contexto da prova de conceito.

```
1 createSignedJWS(clientInfo) {
2   const headers = {
3     typ: "JWT",
4     alg: "PS256", // from JWA (RSASSA-PSS using SHA-256 and MGF1 with SHA
                  -256)
5   };
6
7   // Encode header and payload
8   const encodedPayload = this.base64UrlEncode(JSON.stringify(clientInfo.
    consent));
9
10  // Sign with server private key
11  const signature = this.base64UrlEncode(this.signData(clientInfo.consent))
    ;
12
13  const jws = {
14    payload: encodedPayload,
15    signatures: [
16      {
17        header: headers,
18        signature: clientInfo.signature
19      },
20      {
21        header: headers,
22        signature: signature
23      }
24    ]
25  };
26  return JSON.stringify(jws);
27 }
```


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