Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implantação de um POD server para ecossistema educacional e sua introdução na educação superior em computação

Thiaggo Ferreira Bispo de Souza
Oscar Etcheaverry Barbosa Madureira da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.a Dr.a Germana Menezes da Nóbrega

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implantação de um POD server para ecossistema educacional e sua introdução na educação superior em computação

Thiaggo Ferreira Bispo de Souza
Oscar Etcheaverry Barbosa Madureira da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Germana Menezes da Nóbrega (Orientadora)
CIC/UnB

Prof. Dr. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Fernando William Cruz
FGA/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 25 de julho de 2023

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

FS729i

Ferreira, Thiaggo
Implantação de um POD server para ecossistema educacional e sua introdução na educação superior em computação / Thiaggo Ferreira, Oscar Etcheaverry; orientador Germana Menezes. -- Brasília, 2023.
67 p.

Monografia (Graduação - Ciência da Computação) -- Universidade de Brasília, 2023.

1. Projeto Solid. 2. Solid-POD. 3. Ecossistema educacional. 4. Educação em computação superior. 5. Ensino Superior. I. Etcheaverry, Oscar. II. Menezes, Germana, orient. III. Título.

Dedicatória

Dedicamos esse trabalho aos nossos familiares e amigos, principalmente àqueles que nos apoiaram durante os momentos mais difíceis, em especial aos nossos pais e aos nossos amigos mais íntimos, pois esses nos acompanharam desde o início da nossa jornada na UnB até hoje.

Agradecimentos

Agradecemos à nossa orientadora Germana Menezes da Nóbrega, que foi o nosso pilar de apoio durante toda a jornada de produção deste trabalho. Além disso, mostrou-se uma pessoa gentil, comprometida e paciente. Aos nossos colegas, com quem encontramos soluções para nossos problemas em inúmeras discussões nas reuniões com a nossa orientadora. À UnB e a todos os seus profissionais, pois sem eles não teríamos chegado até aqui. Aos professores Marcos Fagundes Caetano e Fernando William Cruz, que, humildemente, aceitaram compor a banca examinadora. E, por fim, mas não menos importante, às nossas famílias e amigos, que estiveram presentes nos altos e baixos dessa jornada e desempenharam um papel crucial como a fagulha de esperança que iluminou esta jornada.

Resumo

Este trabalho tem como objetivo a implantação de um POD server no ecossistema educacional SmartUnB.ECOS e a introdução dos alunos ao uso dos Solid PODs, com foco na descentralização da web. O trabalho é composto por cinco capítulos, que abordam a contextualização do tema, a revisão da literatura, os elementos básicos empregados na execução, a proposta para a solução do problema de pesquisa e a síntese dos resultados. A introdução do POD server pode trazer benefícios para a educação, como a possibilidade de os alunos terem maior controle sobre seus dados e a criação de um ambiente mais colaborativo e personalizado. A criação de aplicativos Solid é mencionada como uma possibilidade de uso dos conhecimentos adquiridos com a implantação do servidor.

Palavras-chave: Projeto Solid, Solid-POD, ecossistema educacional, educação em computação, ensino superior

Abstract

This work aims to implement a POD server in the SmartUnB.ECOS educational ecosystem and introduce students to the use of Solid PODs, with a focus on web decentralization. The work consists of five chapters that address the contextualization of the topic, literature review, basic elements employed in the execution, the proposed solution to the research problem, and the synthesis of results. The introduction of a POD server can bring benefits to education, such as giving students greater control over their data and creating a more collaborative and personalized environment. The creation of Solid applications is mentioned as a possibility for utilizing the knowledge acquired through the server implementation.

Keywords: Solid Project, Solid-POD, Educational Ecosystem, Computer Education, University Education

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

CIC Departamento de Ciência da Computação.

CMS *Content Management Server*.

CSS (1) *Community Solid Server*.

HTML *HyperText Markup Language*.

HTTP *Hypertext Transfer Protocol*.

IRI *Internationalized Resource Identifier*.

LMS *Learning Management Systems*.

LRS *Learning Records Store*.

LTI *Learning Tools Interoperability*.

MIT *Massachusetts Institute of Technology*.

NSS *Node Solid Server*.

PLE *Personal Learning Environment*.

RDF *Resource Description Framework*.

RDFa *Resource Description Framework in Attributes*.

RFC *Request for Comments*.

Solid OIDC *Solid OpenID Connect*.

TCC Trabalho de Conclusão de Curso.

URI *Uniform Resource Identifier*.

URL *Uniform Resource Locator*.

URN *Universal Resource Name*.

W3C *World Wide Web Consortium*.

WWW *World Wide Web*.

Capítulo 1

Introdução

Os sistemas amplamente utilizados na atualidade estão sob o controle de um pequeno grupo de empresas, o que implica na posse dos dados obtidos no consumo dos serviços prestados. A posse desses dados cria diversas oportunidades para as instituições. Porém, também cria, principalmente para os usuários das aplicações, diversos contratempos: a centralização dos dados dificulta a obtenção de uma *Web* integrada pelo usuário, uma vez que cada rede de dados possui níveis diferentes de quantidade e de atualização dos dados e há pouca ou nenhuma granularidade no controle dos dados compartilhados; um exemplo é o caso das redes sociais, em que as amizades têm que ser redeclaradas em cada rede diferente e possuem pouco controle de como e quais informações são compartilhadas com seus amigos online (YEUNG et al., 2011).

No entanto, no início da *Web* cada pessoa tinha o controle sob seus dados, porém, pouco tempo depois, a posse dos dados acabou ficando com as empresas, o que acabou trazendo riscos à segurança e à privacidade de seus usuários. Com a centralização dos dados há um maior risco de uma quantidade massiva de dados serem corrompidos ou perdidos (ALABDULWAHHAB, 2018). Como solução para as questões levantadas pela *Web* centralizada o criador da *World Wide Web* propôs um retorno à *Web* descentralizada: "A proposta é, então, trazer de volta a ideia da *Web* descentralizada. Trazer de volta o poder para as pessoas." (BERNERS-LEE, 2016). Como uma solução aos problemas levantados anteriormente pela *Web* centralizada, a *Web* descentralizada permite uma maior integração dos serviços, maior controle do usuário sob a localidade dos seus dados, maior granularidade no controle dos dados compartilhados e menor risco de invasão de grandes bases de dados (e, portanto, menos responsabilidade por parte das empresas com relação à segurança dos dados do usuário).

Assim, por exemplo, sem a descentralização dos dados, um aluno que ingressou em diversas instituições durante a sua trajetória educacional não terá seus dados educacionais sob sua posse, estando eles sobre a posse de cada instituição pela qual passou. Isso

dificulta o aproveitamento de informações coletadas durante a trajetória educacional de cada indivíduo, pois com os dados sob a posse de cada instituição, é impossível obter uma visão unificada e abrangente do seu progresso acadêmico.

No entanto, com a adoção de uma abordagem de *Web* descentralizada, em que cada usuário tem maior controle sobre seus próprios dados, seria viável a criação de uma espécie de "passaporte educacional" digital. Nesse modelo, o aluno seria o detentor dos seus registros educacionais, que poderiam ser compartilhados de forma seletiva com diferentes instituições e empregadores ao longo de sua vida.

Essa maior integração e portabilidade dos dados educacionais traz inúmeros benefícios para o estudante. Primeiramente, permitiria uma melhor análise de seu desempenho ao longo do tempo, identificando pontos fortes e fracos e possibilitando o desenvolvimento de estratégias personalizadas de aprendizado. Além disso, facilitaria o reconhecimento de créditos e habilidades adquiridas em diferentes instituições, o que poderia agilizar processos de transferência e reduzir o tempo necessário para completar um determinado curso.

Outra vantagem da descentralização dos dados é a possibilidade de o aluno explorar diferentes abordagens de ensino e estilos de aprendizagem, pois teria acesso a um conjunto mais amplo de informações sobre as opções disponíveis em diferentes instituições. Isso contribuiria para uma formação educacional mais abrangente e adaptada às preferências e necessidades individuais de cada estudante.

Em suma, a *Web* descentralizada oferece a oportunidade de reverter a centralização dos dados e devolver o poder aos indivíduos, possibilitando o uso mais eficiente das informações coletadas ao longo da trajetória educacional de cada aluno. Essa abordagem promissora pode não apenas melhorar a experiência de aprendizado, mas também abrir caminho para uma sociedade mais informada, flexível e capacitada.

A fim de suprir a necessidade de uma *Web* descentralizada, foi criado o protocolo *Solid* (PROJECT, [s.d.]). Esse protocolo propõe conceder maior controle dos dados ao usuário, fornecendo recipientes denominados *Pods*, que são armazenados em servidores *Solid Pod* (ou *Solid Pod servers*, *Pod server* ou *Solid server*), onde o controle de acesso é delegado ao próprio usuário. Os servidores *Solid Pod* são considerados super-recipientes.

1.1 Contextualização

O projeto SmartUnB.ECOS é uma iniciativa de criação de um ecossistema educacional digital para a comunidade universitária com o objetivo de oferecer interoperabilidade entre ferramentas de comunicação e educação, promovendo assim a interação social e o processo de aprendizagem. Esse projeto funciona como uma "caixa de areia" (*sandbox*)

para a execução de pesquisas por meio dos TCCs dos alunos que integram os 3 cursos ofertados pelo Departamento de Ciência da Computação - Bacharelado em Ciência da Computação, Licenciatura em Computação e Engenharia de Computação; ou seja, esse sistema pode ser utilizado tanto como fonte de estudos para o Trabalho de Conclusão de Curso (TCC), quanto pode ser utilizado como um sistema a ser ampliado, com o desenvolvimento de novas funcionalidades (NóBREGA; SILVA; SILVA, 2022).

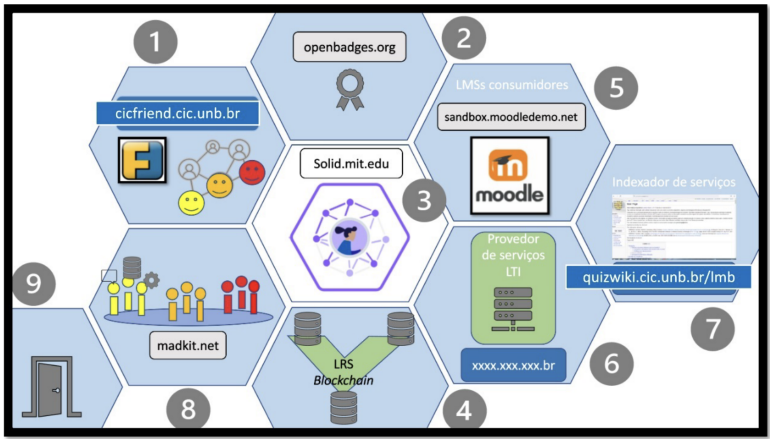

Figura 1.1: Estrutura do SmartUnB.ECOS (NóBREGA; SILVA; SILVA, 2022).

Os elementos que compõem o ecossistema SmartUnB.ECOS, mencionados na Fig. 1.1 são os seguintes (como descritos em (NóBREGA; SILVA; SILVA, 2022)):

1. Rede Social Descentralizada, instância de Friendica [1], implantada para a comunidade do departamento;
2. Plataforma de gestão de *Badges* digitais, para gerir o ciclo de vida de medalhas digitais;
3. Pod *server*, para armazenamento distribuído de registros de aprendizagem - objeto de trabalho desta monografia;
4. LRS (do inglês *Learning Records Store*) em *blockchain*, para armazenamento distribuído de registros de aprendizagem;
5. LMS (do inglês *Learning Management Systems*), consumidores LTI (do inglês *Learning Tools Interoperability*);
6. Provedor LTI de serviços dedicados, ferramentas ou serviços específicos para utilização em disciplinas;
7. Repositório de Recursos Educacionais, que provê indexação semântia dos serviços dedicados do provedor LTI, bem como recursos da produção nacional e internacional;
8. Sociedade de Assistentes Artificiais, agentes benevolentes e autônomos que se comunicam e cooperam em prol da saúde acadêmico do corpo discente;
9. *Learning Companion* ou portal personalizado, interface do ecossistema dotada de mecanismo de personalização.

[1] <https://friendi.ca>

1.2 Motivação e Justificativa

A *Web* tem evoluído desde sua criação. Inicialmente proposta por Tim Berners-Lee como *Web* 1.0 (a *Web* dos documentos, no qual havia uma maior descentralização da *Web*), evoluiu para *Web* 2.0 (a *Web* das pessoas, iniciando um processo de centralização), chegando na *Web* 3.0 (a *Web* dos dados, que está no processo de implementação e que propõe redescentralização dos dados)(HIREMATH; KENCHAKKANAVAR, 2016). Em detalhes, a *Web* 3.0 surge como outra proposta de Tim Berners-Lee, que através do RDF (*Resource Description Framework*) proporciona a utilização da *Web* semântica, trazendo independência dos dados e proporcionando a descentralização do dados da *Web* (NUPUR, 2014). Além de propor a *Web* 3.0, Tim Berners-Lee está a frente do projeto Solid[2] (que busca descentralizar os dados do usuário) e é co-fundador da empresa Inrupt.com (empresa que usa, promove e ajuda as pessoas a desenvolverem a plataforma Solid)[3]. Assim, é necessário que em um contexto universitário exista uma forma de capacitar os alunos a engatinhar no início desse novo paradigma.

Além de capacitar os alunos a utilizar uma *Web* descentralizada, este trabalho também ajudará de forma efetiva no desenvolvimento do ecossistema ecossistema SmartUnB.ECOS. Pois, o Solid Pod, uma das soluções de descentralização da *Web*, que será objeto de estudo deste trabalho, é uma parte fundamental para esse ecossistema.

1.3 Problema de Pesquisa

No intuito de adicionarmos o armazenamento descentralizado ao SmartUnB.ECOS e incentivar o estudo e o contato dos discentes com a descentralização da *Web*, definido o seguinte problema de pesquisa:

> Como implantar um Pod *server* no ecossistema educacional SmartUnB.ECOS e oferecer aos alunos uma forma de iniciar seus estudos na *Web* descentralizada, com foco nos Solid Pods?

Observa-se que não se busca somente a implantação do Pod *server* no ecossistema educacional, mas também a introdução dos discentes ao uso dos Solid Pods. Isso porque a ideia da descentralização da *Web* pode ser implementada de várias formas e o Solid é uma especificação recente. Responder a esse problema de pesquisa é um passo importante para tornar o ecossistema descentralizado e permitir que os alunos estudem e tragam inovação a essa tecnologia que promete empoderar os usuários da *Web*.

[2] <https://solidproject.org/TR/protocol>
[3] <https://www.w3.org/People/Berners-Lee/>

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral desse trabalho é implementar um Pod *server* no ecossistema SmartUnB.ECOS e confeccionar um material para conduzir os discentes que virão a estudar o ecossistema a utilizar as funcionalidades dessa tecnologia. Com isso, os estudantes devem ter uma compreensão da proposta de descentralização da *Web*, ser capazes de compreender o funcionamento dos Solid Pods e adicionar novas funcionalidades.

1.4.2 Objetivos Específicos

O objetivo geral do trabalho pode ser decomposto nos seguintes objetivos específicos:

- Investigar implementações de Solid Pod *servers* da comunidade;
- Implantar no ecossistema SmartUnB.ECOS um Pod *server*;
- Confeccionar material para introduzir os estudantes ao tema da descentralização da *Web*;
- Confeccionar material para introduzir os estudantes aos Solid Pods;

1.5 Estrutura deste documento

Este documento é composto de 5 capítulos que são estruturados da seguinte forma:

- Neste capítulo (Capítulo 1) aborda-se os seguintes assuntos: a contextualização do tema abordado nesse trabalho; a motivação para realizar o trabalho; a definição para o problema de pesquisa; os objetivos do trabalho; a metodologia utilizada para abordar o problema de pesquisa e a estrutura do documento;
- No Capítulo 2 é feita uma revisão da literatura, abordando os conceitos preliminares abordando as importantes iniciativas no âmbito da descentralização da *Web*, da criação de aplicativos Solid e dos ecossistemas educacionais;
- O Capítulo 3 se concentra nos elementos básicos que foram empregados em sua execução. Para tal, é apresentada uma base teórica sólida e as tecnologias utilizadas para alcançar os objetivos propostos.
- No Capítulo 4 é apresentada a proposta para a solução do problema de pesquisa.
- No Capítulo **??** é feita a síntese dos resultados, trata das contribuições deste trabalho para a área de pesquisa e

Capítulo 2

Revisão de Literatura

O objetivo deste capítulo é apresentar as referências teóricas que embasaram o estudo. O capítulo está dividido em conceitos preliminares - conceitos básicos e fundamentais necessários para o entendimento do tema abordado - e trabalhos relacionados - que apresenta trabalhos que foram desenvolvidos anteriormente sobre o tema abordado.

2.1 Conceitos Preliminares

2.1.1 *Lifelong Learning*

O *Lifelong Learning* significa "Aprendizagem ao Longo da Vida". É a construção contínua de habilidades e conhecimentos ao longo da vida de um indivíduo (LAAL; SALAMATI, 2012). Isso não apenas aumenta a inclusão social, a cidadania ativa e o desenvolvimento pessoal, mas também a competitividade e a empregabilidade. Esse conceito engloba várias formas de educação, como a aprendizagem formal, informal e autodirigida.

Atualmente, o modelo de economia se baseia no conhecimento, o que gera uma crescente pressão para que as pessoas se mantenham atualizadas. Com as tecnologias avançando em uma velocidade cada vez maior, surge a necessidade de aprimorar constantemente os conhecimentos para se adaptar à vida moderna. Essa demanda se reflete não apenas na busca por profissionais capacitados, mas também na necessidade de que as pessoas desenvolvam habilidades e competências para acompanhar as mudanças do mundo atual.

O *Lifelong Learning* traz inúmeros benefícios para os indivíduos e a sociedade como um todo. Alguns desses benefícios incluem:

- **Aprimoramento das habilidades mentais:** O aprendizado contínuo ajuda a manter a mente afiada e melhora a memória. Estudos mostram que pessoas com

maior nível de educação têm menor probabilidade de desenvolver demência na velhice.

- **Aumento da confiança:** Ao se envolver em processos de aprendizagem ao longo da vida, as pessoas ganham confiança em sua capacidade de aprender e compartilhar conhecimentos. Isso fortalece a autoestima e a percepção de si mesmas.
- **Desenvolvimento das habilidades interpessoais:** Oportunidades de socialização proporcionadas pelo aprendizado contínuo contribuem para o aprimoramento das habilidades interpessoais. Ao interagir com outras pessoas e compartilhar conhecimentos, estabelecemos relacionamentos mais sólidos e enriquecedores.
- **Expansão das oportunidades de carreira:** O *Lifelong Learning* não apenas aprimora as habilidades existentes, mas também oferece a oportunidade de aprender novas habilidades e ampliar as possibilidades de crescimento profissional.
- **Melhoria na habilidade de comunicação:** O processo de aprendizagem geralmente envolve o aprimoramento das habilidades de leitura, audição e escrita, que são essenciais para a comunicação efetiva. O desenvolvimento dessas habilidades permite escrever cartas comerciais, elaborar relatórios de marketing, fazer apresentações e se comunicar com mais eficácia em diferentes contextos.

Em resumo, o *Lifelong Learning* é fundamental no mundo em constante mudança em que vivemos. Ele oferece oportunidades de crescimento pessoal, aprimoramento profissional e adaptação às demandas da sociedade contemporânea. Portanto, devemos abraçar os benefícios do aprendizado contínuo e aproveitar ao máximo o tempo que temos neste planeta.

Este trabalho se encaixa no contexto do *Lifelong Learning* por trazer uma abordagem que objetiva concentrar os dados do usuário em um único lugar, permitindo armazenar toda a bagagem educacional ao longo da vida do estudante. Além disso, busca abrir os olhos do estudante para um novo paradigma da *Web*, incentivando a criatividade e possibilitando que o estudante considere novas abordagens além das tradicionais na *Web*.

No âmbito do *Lifelong Learning*, reconhece-se a importância de um ambiente que promova a continuidade do aprendizado ao longo da vida. O armazenamento centralizado dos dados educacionais do estudante, proposto neste trabalho, proporciona uma plataforma que facilita o acesso e a organização dessas informações, promovendo a reflexão sobre o próprio percurso educacional e estimulando a busca por novos conhecimentos.

Ademais, ao introduzir um novo paradigma da *Web*, este trabalho busca incentivar a criatividade e a experimentação por parte dos estudantes. Ao explorar novas abordagens

e possibilidades além das já existentes, os estudantes são encorajados a expandir seus horizontes e considerar novas formas de interação, colaboração e aprendizado.

Nesse sentido, a proposta deste trabalho contribui para o conceito de *Lifelong Learning* ao tratar de uma ferramenta que auxilia os estudantes a gerenciar sua bagagem educacional e explorar novos caminhos na aprendizagem ao longo da vida.

2.1.2 Universal Resource Identifier

O termo *Uniform Resource Identifier* (URI) foi descrito em 1998, no *Request for Comments* (RFC) 2396 (BERNERS-LEE; MASINTER; FIELDING, 1998). Seu objetivo principal era fornecer uma maneira simples e extensível de identificar recursos na *Web* utilizando caracteres ASCII. O URI unificou e atualizou as definições de *Uniform Resource Locator* (URL) (BERNERS-LEE; MASINTER; MCCAHILL, 1994) e *relative URL* (FIELDING, 1995) . Ele pode ser dividido em *Uniform Resource Locator* (URL), que especifica a localização de um recurso e como acessá-lo, e *Universal Resource Name* (URN) (MOATS, 1997), que especifica o nome ou identidade de um recurso. Além disso, um URI pode ser uma combinação desses dois componentes.

Ademais, é importante destacar que o URI é amplamente utilizado na *Web* para identificar e acessar recursos, como páginas da *Web*, imagens, documentos e serviços. Ele desempenha um papel fundamental na infraestrutura da Internet, permitindo que os usuários acessem informações e recursos por meio de navegadores e outros aplicativos.

Além disso, o URI é flexível e permite a criação de identificadores únicos para uma ampla variedade de recursos, independentemente de sua localização física. Isso é especialmente relevante na era da computação em nuvem, onde os recursos podem estar distribuídos em vários servidores e sistemas.

Em resumo, o URI é um conceito essencial na *Web*, fornecendo um sistema de identificação padronizado para recursos. Sua combinação de *Uniform Resource Locator* (URL) e *Universal Resource Name* (URN) permite especificar a localização e o nome/identidade de um recurso, proporcionando uma base sólida para a navegação e o acesso eficiente aos recursos da *Web*.

2.1.3 Identificador de Recursos Internacionalizado

O termo *Internationalized Resource Identifier* (IRI) foi criado em 1969 e está documentado no RFC 3987 (DüRST; SUIGNARD, 2005). Ele representa um avanço em relação ao *Uniform Resource Identifier* (URI), sendo uma extensão que permite a inclusão de uma ampla variedade de caracteres do Conjunto de Caracteres Universal (Unicode / ISO

10646). Um IRI pode ser mapeado para um URI, possibilitando sua substituição quando necessário (DüRST; SUIGNARD, 2005).

A criação do IRI surgiu como resposta a uma questão filosófica subjacente: enquanto os URIs são facilmente legíveis por máquinas, eles também são amplamente utilizados em materiais destinados a seres humanos, como panfletos e propagandas. No entanto, os URIs geralmente são limitados a caracteres ASCII e são predominantemente baseados no idioma inglês. Essa abordagem cria barreiras de acessibilidade, favorecendo aqueles que estão familiarizados com o inglês, ao passo que os IRIs abrangem um conjunto mais amplo de caracteres, contemplando assim mais línguas e culturas.

É importante destacar que a adoção de IRIs promove a inclusão e a diversidade na *Web*, permitindo que diferentes idiomas e caracteres sejam utilizados de forma equivalente na identificação e localização de recursos. Isso tem um impacto significativo na acessibilidade global, na internacionalização de sites e na representação adequada da diversidade cultural.

Em resumo, os IRIs são uma extensão dos URIs que abrangem um conjunto mais amplo de caracteres e possibilitam a inclusão de diferentes idiomas e culturas na identificação de recursos da *Web*. Sua criação visa superar as limitações dos URIs em relação à representação multilíngue, promovendo uma *Web* mais inclusiva e acessível para usuários de todo o mundo.

2.1.4 *Web* Semântica

A *Web* semântica é uma extensão da *Web* 1.0, a *Web* dos documentos, que suporta a mudança para a *Web* 3.0, a *Web* dos dados (BRASIL, 2011) através do pilar do RDF. Exatamente a *Web* semântica procura uma forma de padronizar os significados dos dados, pois os dados podem ser compartilhados e reutilizados por outras plataformas, facilitando o uso de aplicativos (BRASIL, 2011), ou seja, permite que os dados utilizado por uma aplicação não sejam desperdiçados ou escondidos, mas utilizados por uma aplicação que o usuário permitiu, evitando o recalculo de dados da aplicação.

Essa tecnologia vem crescendo e possui diferentes domínios de aplicações (JANEV; VRANEš, 2011): Integração de dados; Pesquisa semântica; Descoberta de conteúdo semântico; Interfaces de linguagem natural; Integração de serviços. Também existem usos específicos na agricultura (MOREIRA et al., 2015) , assim como usos no campo da internet das coisas (DRURY et al., 2019).

O Solid se enquadra na *Web* Semântica ao propor o armazenamento de dados nos Pods de forma estruturada e com anotações semânticas, o que permite que os usuários e as aplicações possam acessar, manipular e integrar os dados de forma mais eficiente e segura.

2.1.5 *Linked Data*

Também conhecido como Dados Interligados, é um conceito, de potencial inovador, introduzido por Tim Berners-Lee, diretor do *World Wide Web Consortium* (W3C), em 2006. Na era da *Web* semântica, o *Linked Data* desempenha um papel crucial ao permitir a conexão e compartilhamento de dados estruturados de forma padronizada. Isso viabiliza a descoberta, combinação e utilização eficiente das informações.

Essa abordagem revolucionária utiliza tecnologias fundamentais da *Web*, como HTML, RDF e URIs, para conectar dados legíveis por máquinas através de *hyperlinks*. Isso forma uma rede de dados interconectados que pode ser navegada. Para garantir sua aplicação correta, Tim Berners-Lee estabeleceu os Princípios do *Linked Data* (BIZER; HEATH; BERNERS-LEE, 2009) (BERNERS-LEE, 2006):

1. Utilização de URIs como identificadores únicos para recursos, garantindo identificação precisa.
2. Uso de URIs baseadas em HTTP, que permitem busca e recuperação dos dados por meio de consultas semânticas eficazes.
3. Disponibilização de informações significativas quando uma URI é acessada, seguindo os padrões SPARQL e RDF. Isso permite que máquinas obtenham conhecimento sobre os recursos e suas relações.
4. Incorporação de links para outras URIs relacionadas, facilitando a descoberta de informações adicionais e a navegação entre os dados interligados.

O *Linked Data* traz benefícios significativos, promovendo a interoperabilidade e enriquecimento das informações. Além disso, capacita consultas semânticas avançadas por meio do SPARQL, permitindo extração de conhecimento precisa e eficiente. Apesar de suas vantagens, o Linked Data enfrenta desafios, como a garantia da qualidade e consistência dos dados, bem como a proteção da privacidade e segurança das informações compartilhadas.

No entanto, o impacto potencial do *Linked Data* é notável. Ele impulsiona a inteligência artificial, permitindo a criação de sistemas mais inteligentes e personalizados. Além disso, a conexão de ideias e informações provenientes de diferentes domínios aprimora a descoberta de conhecimento.

Em suma, o *Linked Data* transforma a *Web* em uma vasta rede de dados interligados, impulsionando a *Web* semântica e viabilizando a criação de aplicativos e serviços mais inteligentes e significativos. O projeto Solid se encaixa nessa estrutura, promovendo o uso de tecnologias relacionadas, como o RDF.

2.2 Trabalhos Relacionados

Nessa subseção do artigo serão exploradas as pesquisas e descobertas relevantes que moldaram o campo até o momento, fornecendo uma base sólida para o desenvolvimento do presente estudo.

É importante citar que os trabalhos foram retirados de fontes das áreas de Informática na Educação e Computação. São elas: *Journal of Education and Learning*, IEEE *Transactions* on *Industrial Informatics, WWW'16 Companion*,

2.2.1 Análise da descentralização da internet

As grandes redes sociais, muitas vezes, são de propriedade e gerenciadas apenas por uma única empresa, chama-se redes sociais centralizadas, e com os dados que os usuários fornecem gratuitamente, essas empresas são capazes de monetizar os dados do usuário, assim como os dados derivados dos dados do usuário. E para solucionar esse e diversos outros problemas das redes sociais centralizadas foi, pro que houvesse uma descentralização das redes sociais. Assim, (BAHRI; CARMINATI; FERRARI, 2018) fizeram um trabalho, intitulado como "Decentralized privacy preserving services for Online Social Networks", que faz uma análise dos potenciais problemas de privacidade em redes sociais descentralizadas.

Esse artigo (BAHRI; CARMINATI; FERRARI, 2018) destaca 3 principais problemas para a descentralização dos dados do usuário: Armazenamento e replicação de dados; Controle de acesso; E gerenciamento de conteúdo e de identidade falsos. O primeiro problema trata do armazenamento e sobre a replicação de dados entre os entes que o usuário confia, onde é proposto 2 soluções, os entes de armazenamento possuem entes de armazenamento de confiança, e o usuário pode escolher dentre diversos ente de armazenamento. Para o controle de acesso, as soluções podem ser divididas em 2 grupos: as que são baseadas de criptografia, onde todas as informações são abertas e criptografadas e somente quem obtiver a chave publica poderá ter acesso a esses dados, e as que não são baseadas de criptografia, que dentre as várias soluções propostas destaca a que permite somete o acesso de alguns usuários (que devem estar logados). Já para o gerenciamento de conteúdo e de identidade falsos é descrito 2 abordagens, que não são alternantes: identificação de usuários falso e identificação de usuários com prestígio.

O artigo (BAHRI; CARMINATI; FERRARI, 2018), relaciona com esta monografia, pois apresenta uma outra forma de ver sobre a descentralização de dados, seu problemas e soluções. Bem como, uma divergência na parte do armazenamento e replicação de dados e uma pequena semelhas na parte de controle de acesso.

O artigo "MobiTribe: A Peer-to-Peer Content Sharing System for Mobile Social Networks" (THILAKARATHNA et al., 2014) propõe uma abordagem *peer-to-peer* para

o compartilhamento de conteúdo em redes sociais móveis. O sistema, chamado MobiTribe, permite que os usuários compartilhem seu próprio conteúdo com outros usuários, registrando o conteúdo com o *Content Management Server* (CMS) imediatamente após a criação e anunciando um link para o conteúdo para os usuários que se inscreveram para receber atualizações do criador de conteúdo através de um serviço de rede social, como o Facebook [1]. Em seguida, cada dispositivo inscrito decide se deve pré-buscar o conteúdo com base na demanda prevista e informa o CMS de sua intenção de baixar o conteúdo. O CMS decide se esses dispositivos, com base nos padrões de disponibilidade, fornecem disponibilidade suficiente do conteúdo para os outros consumidores sob demanda.

O MobiTribe é uma outra maneira de abordar o acesso distribuído de conteúdo armazenado em usuário, assim como o projeto Solid de Tim Berners-Lee. Ambos os projetos, MobiTribe e Solid, têm como objetivo descentralizar o acesso a conteúdo armazenado em usuários, permitindo que os usuários controlem seus próprios dados e compartilhem esses dados com outros usuários de forma descentralizada. Enquanto o MobiTribe se concentra no compartilhamento de conteúdo em redes sociais móveis, o Solid tem como objetivo descentralizar a Web como um todo, permitindo que os usuários controlem seus próprios dados em todos os aspectos da Web.

Além disso, ambos os projetos são importantes para garantir a privacidade e segurança dos dados dos usuários, bem como para criar uma Web mais descentralizada e democrática. Ambos os projetos têm o potencial de mudar a forma como os usuários interagem com a Web e com outros usuários, permitindo que eles controlem seus próprios dados e decidam como compartilhá-los.

No entanto, é importante notar que existem diferenças significativas entre os dois projetos. Enquanto o MobiTribe usa uma abordagem peer-to-peer para distribuir conteúdo entre os dispositivos móveis dos usuários, o Solid usa uma abordagem baseada em servidores pessoais para armazenar e compartilhar dados entre aplicativos e serviços. Além disso, o Solid usa padrões da Web Semântica para permitir que os usuários compartilhem dados entre aplicativos e serviços, enquanto o MobiTribe usa um CMS para gerenciar o compartilhamento de conteúdo entre os usuários.

Em resumo, tanto o MobiTribe quanto o Solid são projetos importantes que abordam a questão do acesso distribuído de conteúdo armazenado em usuário. Ambos os projetos estudam mudanças na forma como os usuários interagem com a Web e com outros usuários, permitindo que eles controlem seus próprios dados e decidam como compartilhá-los.

[1] <https://www.facebook.com/>

2.2.2 A Mudança na Educação com NFTs, DAOs, *Web* 3.0 e o Metaverso

O trabalho intitulado "*Tokens* não fungíveis, organizações autônomas descentralizadas, *Web* 3.0 e o metaverso na educação: Da universidade à metaversidade" (SUTIKNO; AISYAHRANI, 2023) (em Português) aborda questões relevantes sobre como as tecnologias emergentes da *Web* 3.0 podem transformar a educação no contexto do ensino superior. O estudo explora conceitos como *tokens* não fungíveis, organizações autônomas descentralizadas, *Web* 3.0 e metaverso, examinando suas aplicações tanto no ambiente educacional tradicional quanto no virtual.

Ao analisar o impacto dessas tecnologias, o trabalho discute como elas podem revolucionar a maneira como estudantes e professores armazenam, compartilham, validam e interagem com seus dados e conteúdos educacionais. Além disso, destaca as possibilidades oferecidas pelas tecnologias da *Web* 3.0, como aprendizagem imersiva e colaborativa, que podem enriquecer a experiência educacional.

Uma contribuição importante do trabalho é a proposição do termo "metaversidade", que se refere a uma instituição de ensino superior reconstruída como um gêmeo digital no metaverso, fazendo uso da realidade virtual ou aumentada. Essa abordagem inovadora pode criar ambientes de aprendizagem mais envolventes e interativos, oferecendo alternativas às salas de aula físicas ou virtuais tradicionais.

O estudo relaciona-se diretamente com o tema de implantação de um Pod *server* para o ecossistema educacional, pois demonstra como as tecnologias da *Web* 3.0 podem fornecer a infraestrutura necessária para uma experiência educacional descentralizada e personalizada. Ele ressalta a importância de proporcionar maior autonomia, privacidade, transparência e confiança aos estudantes e professores no contexto da educação online. Além disso, apresenta *insights* valiosos sobre como o metaverso pode ser explorado como um ambiente complementar ou substituto das salas de aula, oferecendo possibilidades de aprendizagem mais imersivas e interativas.

2.2.3 O uso do *Solid* em hospitais

Uma das aplicações do solid pod em projetos de larga escala são os sistemas de hospitais. Isso, pois a estrutura de funcionamento dos dados de um hospital oferece uma estrutura ideal para aplicar o solid pod, essa essa estrutura pode ser observada com os seguintes aspectos: os usuários com seu dados médicos em seu pod; a capacidade de um paciente dar, ou retirar, a permissão que um médico possa alterar os seus dados pessoais deste paciente; a possibilidade de de mudar de hospital sem aumentar a burocracia. Esses

benefícios são descritos por Ghayvat, Gope e Sharma(GHAYVAT et al., 2022), e pela matéria da BBC[2].

A matéria da BBC, descreve que foi feito um experimento iniciado pelo NHS (*National Health Service* — Serviço Nacional de Saúde do Reino Unido) em em 2020 na localidade da Grande Manchester. É explicado que o foco desse experimento são os cuidadores de pessoas com demência, ou qualquer profissional de saúde que possam tratá-los, as informações guardadas vão desde os exames médicos até informações sobre como acalmar o paciente. E após enfatizar a utilização de pods pode melhorar a troca de informações entre os hospitais, é descrito que se for um sucesso essa será uma forma revolucionária de lidar com os dados dos sistemas de hospitais. Também é possível ver uma versão da implementação em vídeo, onde é possível exemplos de de autorização de acesso de dados.

Ainda assim, existem cuidados necessários para a fazer a implementação(GHAYVAT et al., 2022): Um médico malicioso pode ter acesso a recursos por uma autorização indevida e escapar sem ser detectado; Um agente malicioso com poder para ler ou alterar; E que o usuário alterar os seu próprio dados médicos. Contudo, também é apresentado que esses problemas podem ser facilmente resolvidos com a implementação de tecnologia adequadas.

2.2.4 Demonstração da plataforma Solid

O artigo (MANSOUR et al., 2016) apresenta uma demonstração da plataforma Solid, que é uma plataforma descentralizada para aplicativos da *Web* social. O artigo descreve a arquitetura da plataforma Solid, que é baseada em tecnologias da *Web* Semântica e em padrões do W3C. A plataforma Solid especifica todos os protocolos necessários para a comunicação entre aplicativos e servidores, incluindo autenticação, comunicação entre aplicativo e servidor e comunicação entre servidores.

O artigo apresenta uma série de aplicativos que demonstram a interoperabilidade e o controle de acesso por meio de aplicativos Solid. Os aplicativos incluem gerenciamento de contatos, gerenciamento de eventos, edição colaborativa de artigos acadêmicos, entre outros. Além disso, o artigo destaca a facilidade de implementação de recursos sociais em aplicativos Solid e o uso de métodos HTTP/1.1 para interação entre aplicativos e pods.

Este artigo é relevante para o tema deste trabalho, pois fornece informações sobre a implementação e uso de aplicativos Solid, que podem ser úteis para a criação de um tutorial sobre como usar o servidor Solid. Além disso, o artigo destaca a importância da interoperabilidade e do controle de acesso em aplicativos Solid, que são aspectos importantes a serem considerados na implementação de um servidor Solid para fins educacionais.

Outro ponto importante abordado no artigo é a portabilidade de dados entre servidores Solid. Isso significa que os usuários podem armazenar seus dados em um servidor Solid

[2]<https://www.bbc.com/news/technology-54871705>

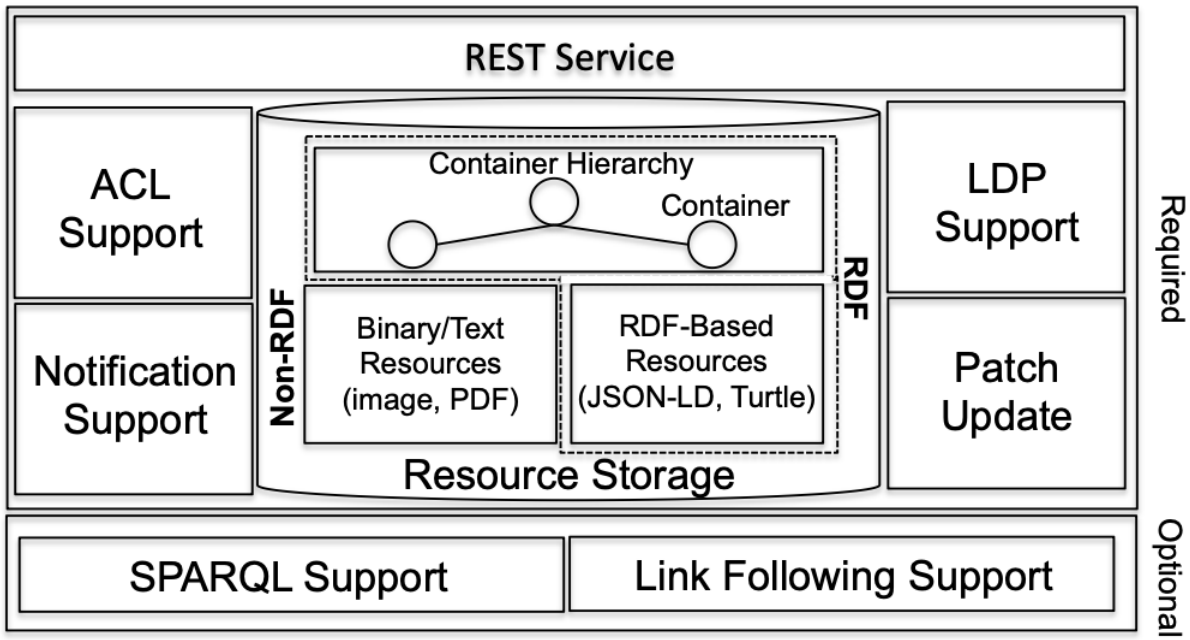

Figura 2.1: Visão geral de um Pod *server* (MANSOUR et al., 2016)

e, se desejarem, transferi-los para outro servidor Solid sem perder nenhum dado. Essa característica é importante para a privacidade e a segurança dos dados dos usuários, além de permitir que os usuários tenham mais controle sobre seus dados.

2.2.5 Descentralização da *Web* e PLEs como sistemas sociotécnicos

O presente trabalho tem como objetivo a implantação de um servidor Solid Pod em um ecossistema educacional e sua introdução na educação superior em computação. Nesse contexto, é relevante considerar o estudo realizado por Kamakshi Rajagopal sobre Personal Learning Environments (PLEs) como sistemas sociotécnicos (RAJAGOPAL, 2023).

No estudo mencionado anteriormente, é discutido o conceito de PLEs, que são ambientes de aprendizagem personalizados que permitem que os alunos gerenciem seus próprios processos de aprendizagem, selecionando e combinando ferramentas e recursos de acordo com suas necessidades e preferências. A abordagem sociotécnica considera a relação entre os aspectos sociais e técnicos dentro desses ambientes, destacando como os elementos tecnológicos interagem com os fatores sociais, culturais e pedagógicos para promover uma aprendizagem mais efetiva.

Ao implantar o servidor Solid Pod no ecossistema educacional, pretende-se fornecer uma infraestrutura que permita aos alunos criar e gerenciar seus próprios PLEs. Isso

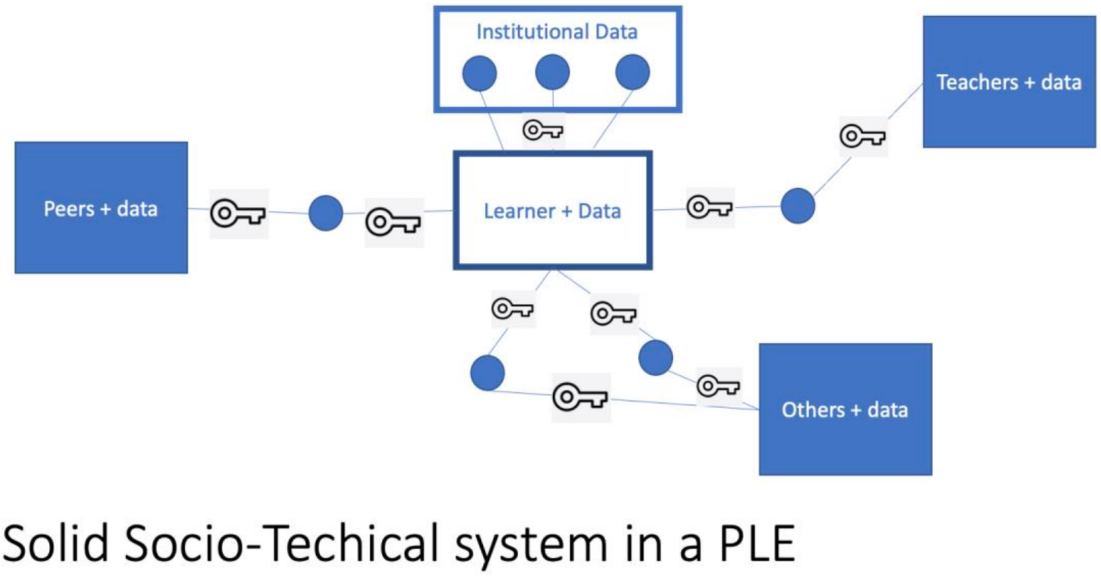

Figura 2.2: Modelo de sistema sociotécnico decentralizado em um PLE (RAJAGOPAL, 2023)

permite que eles personalizem seu ambiente de aprendizagem, selecionem as ferramentas e recursos mais adequados às suas necessidades e compartilhem informações de maneira colaborativa. Além disso, a abordagem sociotécnica considera a interação entre os usuários, as ferramentas e a infraestrutura tecnológica, o que é fundamental para criar um ambiente educacional mais adaptativo e centrado no estudante.

Portanto, o estudo de Rajagopal sobre PLEs como sistemas sociotécnicos fornece uma base teórica importante para a compreensão dos aspectos sociais, culturais e tecnológicos envolvidos na implantação do servidor Solid Pod. Essa perspectiva contribui para o desenvolvimento de um ecossistema educacional mais dinâmico e personalizado, que promove uma aprendizagem mais significativa e engajadora na educação superior em computação.

2.2.6 *Timeline* do Projeto Solid: Evolução e Marcos Históricos

Para melhor entender o contexto, a evolução e o papel do projeto Solid na *Web*, pode-se traçar uma linha do tempo, começando a partir da criação da *World Wide Web* (WWW) em 1986 (BERNERS-LEE, 1990).

A *Web* foi inventada por Tim Berners-Lee em 1989, ele era um cientista na empresa CERN. A ideia era combinar computadores, redes de dados e hipertexto em um sistema global de informação. Então, desenvolveu o primeiro servidor e navegador *Web* no CERN,

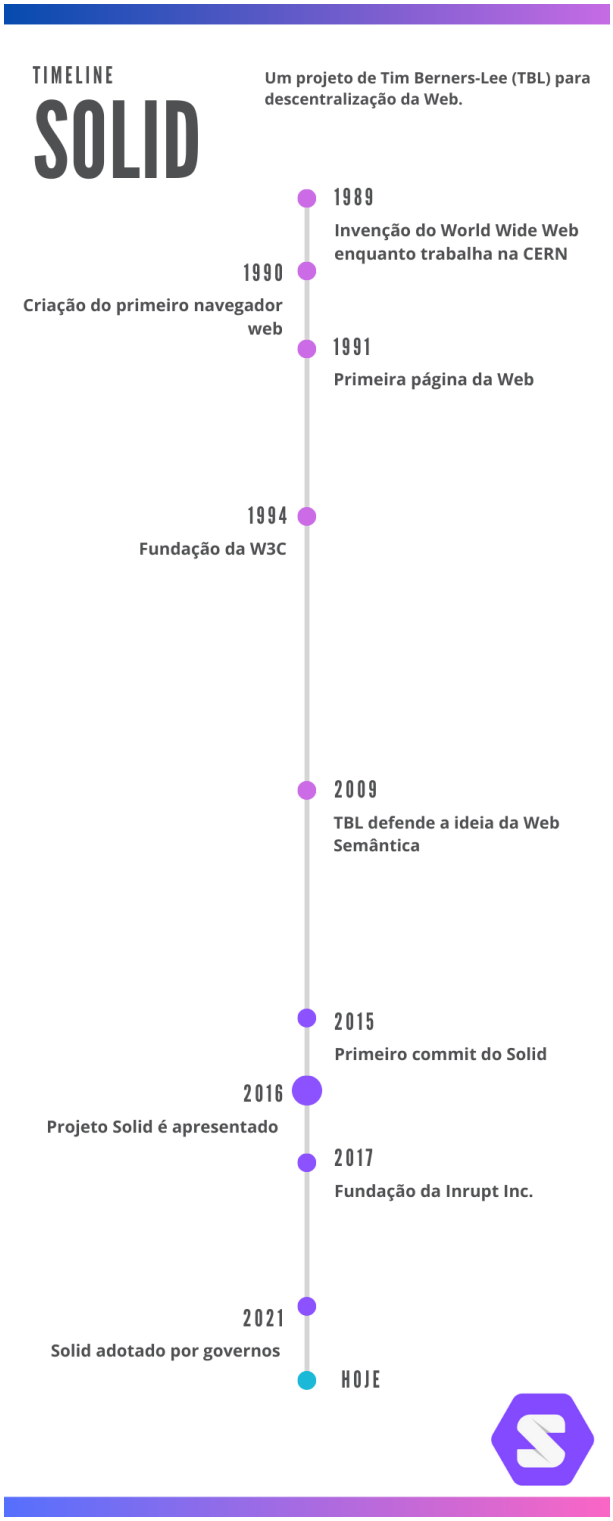

Figura 2.3: *Timeline* do projeto Solid e alguns eventos relacionados

usando um computador NeXT (empresa criada por Steve Jobs em 1985), e lançou o software da *Web* em 1991. A *Web* se espalhou rapidamente pelo mundo, graças à colaboração de outros desenvolvedores e ao lançamento do navegador Mosaic em 1993 (CERN, 2023).

Tim Berners-Lee deixou o CERN para, em 1994, fundar o *World Wide Web Consortium* (W3C). O consórcio foi fundado pois era necessário que alguma organização coordenasse os padrões abertos da *Web*, diante do rápido avanço da tecnologia *Web*. O W3C também foi fundado com a visão de ser uma comunidade global de pares, com presença em diferentes regiões do mundo (HISTORY, 2023).

Em 2009, Tim Berners-Lee publica o artigo (SHADBOLT; BERNERS-LEE; HALL, 2006), defendendo a ideia da *Web* Semântica. Nele, Tim Berners-Lee afirma que a *Web* Semântica é uma extensão da *Web* que visa tornar o conteúdo mais acessível e compreensível para humanos e máquinas. Também menciona uma necessidade tecnológica urgente para o sucesso da *Web* Semântica, assim como a importância dos primeiros adotantes para a adoção dela. Nesse mesmo ano, Tim-Berners Lee faz uma apresentação no TED (BERNERS-LEE, 2009) em que fala do futuro da *Web* e sobre o seu próximo projeto que irá reformular a maneira de usar os dados.

Em 2015, foi dado o primeiro *commit* no repositório do projeto na plataforma de controle de versões GitHub. Em 2016, Tim Berners-Lee apresenta o projeto Solid por meio do artigo (SAMBRA et al., 2016), apesar de já ser um projeto interno do *Massachusetts Institute of Technology* (MIT) anteriormente. O artigo descreve como o Solid usa tecnologias de dados vinculados para fornecer uma estrutura para a criação de aplicativos sociais descentralizados. Ele também explica como o Solid permite que os usuários controlem seus próprios dados, em vez de confiar em grandes empresas de tecnologia para gerenciá-los. Além disso, o artigo discute como o Solid pode ser usado para criar aplicativos sociais que são mais seguros e privados do que os aplicativos tradicionais baseados em *Web*.

Em 2017, a empresa Inrupt Inc. foi fundada por Tim Berners-Lee, com o intuito de prover serviços de nível empresarial baseados no Solid (RENJIFO, 2022) e difundir a tecnologia. Em dezembro de 2021, a Inrupt arrecadou 30 milhões de dólares como financiamento, fechou contratos com governos da Suécia, Argentina e País Basco, e planeja expandir sua presença global nos setores governamental e comercial com os recursos arrecadados (SINGH, 2021).

Capítulo 3

Fundamentos para a proposta

Este capítulo apresenta os fundamentos necessários para compreender a proposta, dividido em 2 seções: Fundamentação conceitual; e Referencial tecnológico. Na fundamentação conceitual será visto os conceitos tóricos para a compreensão da proposta, assim como conceitos que destacam sua relevância. Enquanto no referencial tecnológico será visto a tecnologias que antecederam o projeto Solid, e as tecnologias educacionais usadas para o tutorial na qual este projeto contribui positivamente.

3.1 Fundamentação conceitual

3.1.1 A *Web* descentralizada segundo o projeto Solid

A *Web* descentralizada permite com que todos os usuários possuam maior controle de seus dados (JANSSEN et al., 2021), com uma ID digital global que atuará como uma identidade única exclusiva (STORY et al., 2011). O projeto Solid vai de encontro com esses conceitos e foi proposto por Tim Berners-Lee para retornar o controle dos dados e da privacidade aos usuários e está relacionado com os mesmos objetivos da *Web* também criada por Tim Berners-Lee: formar uma sociedade equitativa, informada e interconectada (PROJECT, 2021).

O projeto Solid permite com que cada usuário guarde seus dados em um banco de dados *Web* acessível e pessoal também chamado de pod. As aplicações usam um protocolo de autenticação para identificar a identidade e os dados do perfil do usuário, assim como links para o pod do usuário, onde estão os dados da aplicação (MANSOUR et al., 2016). Para que esses mecanismos funcionem, é utilizada a ID digital única mencionada acima chamada de WebID; essa ID é um *Uniform Resource Identifier* (URI) HTTP que se refere a um agente (Pessoa, Organização, Grupo, Dispositivo, etc.) e leva a uma documento de perfil que é uma página *Web* que usa uma padronização *Resource Description Framework*

(RDF) (STORY et al., 2011). Para obter uma WebID é necessário que o usuário se registre em um provedor, que normalmente é o seu pod *server*.

Em aplicações Solid, com o uso do WebID, o usuário não precisa se autenticar com o provedor da aplicação, como na maioria das aplicações. Ao invés disso as aplicações podem forçar um falso pedido de autenticação para obter o WebID do usuário a partir do certificado do cliente. A implementação do WebID permite uma gestão global de identidade baseada no conceito de provedores de identidade descentralizados, que quando combinados com o WebID-TLS, possibilita o *single sign-on* em escala *Web*. Em outras palavras, o usuário só precisa fazer o login uma única vez para ter acesso a todas as aplicações que usam o WebID (SAMBRA et al., 2016).

3.1.2 Solid-Pod no ensino superior em computação

Conhecer a *Web* descentralizada é importante para o estudante de computação, pois permite que o estudante tenha contato com um novo paradigma da *Web*, uma vez que a população em geral se encontra acostumada com a *Web* 2.0, que é centralizada. Com isso, o aluno tem contato com novos conceitos, *frameworks* e novas formas de lidar com dados, autenticação e privacidade, além de outros fatores que integram a *Web* descentralizada.

Incorporar esses conceitos à base de conhecimento do estudante, assim como qualquer estudo em campos inovadores, irá prepará-lo melhor para o futuro, pois o estudante estará bem preparado para trabalhar com a *Web* decentralizada que se pode se tomar cada vez mais conta da *Web* e fundar um novo mercado.

Interagindo com esse tema o estudante será capaz de atuar na confecção de novas aplicações voltadas a esse paradigma, compreender as questões éticas e casos de mal uso dos dados dos usuários, como (BOERBOOM, 2020), que acarretaram na proposta da descentralização e será capaz de reutilizar esses conhecimentos para propor ou estudar um novo paradigma para a *Web*, se necessário. Ademais, o estudante poderá ainda integrar a comunidade que desenvolve o projeto Solid, dando mais visibilidade e corpo para o projeto.

De igual importância é a implantação de um Pod *server* local no Departamento de Ciência da Computação (CIC) pois, dessa forma, os estudantes poderão ter mais um elemento do ecossistema proposto em (NóBREGA; SILVA; SILVA, 2022) com o aspecto de uma "caixa de areia"(*sandbox*). Assim, os estudantes poderão, ainda na universidade, desenvolver aplicações Solid compatíveis que podem integrar o ecossistema educacional e expandir o ferramental disponível para trabalhar com os Solid Pods, ou seja, poderão solidificar seus conhecimentos em descentralização e no projeto Solid.

A implantação de um Solid Pod no CIC também permite que mais pesquisas sejam feitas no campo da decentralização por graduandos, integrantes de programas de iniciação

científica, mestrandos e doutorandos. Abre-se a possibilidade de abordagem de temas sobre o uso de Solid Pods como os seguintes: ética no uso de Solid-Pods, segurança da informação, direito e política pública. Um possível assunto a ser tratado como tema de pesquisa em Solid Pods com foco em direitos e segurança da informação é a propriedqde de dados gerados com base nos dados compartilhados.

3.2 Referencial tecnológico

3.2.1 Tecnologias subjacentes ao projeto Solid

Framework de Descrição de Recurso

O *Framework* de Descrição de Recurso (*Resource Description Framework* - RDF) é uma recomendação do *World Wide Web Consortiu*m (W3C) que define uma linguagem para a descrição de recursos (PRUD'HOMMEAUX et al., 2014). O RDF é utilizado para representar e trocar informações estruturadas na *Web*, permitindo que os dados sejam interpretados e processados de forma eficiente por máquinas.

Essa especificação é necessária porque os recursos da *Web* são principalmente direcionados à leitura humana. Por exemplo, os recursos são frequentemente apresentados em formatos como HTML, que são projetados para facilitar a visualização e navegação pelos usuários. No entanto, para que as máquinas possam compreender e utilizar esses dados de forma automatizada, é necessário um formato como o RDF.

Uma das principais vantagens do RDF é sua capacidade de representar informações como um grafo. Nessa estrutura, os recursos são representados como nós e as arestas são os links que conectam esses dados. Essa abordagem permite que diferentes aplicações interpretem e processem os dados de forma consistente, promovendo a interoperabilidade entre sistemas heterogêneos.

Alternativamente, o RDF também pode ser representado como uma tripla, que consiste em um sujeito, um predicado e um objeto. O sujeito representa o recurso em questão, o predicado indica a propriedade do recurso e o objeto é o valor associado a essa propriedade. Por exemplo, uma tripla RDF pode ser composta pelo sujeito "Livro A", o predicado "Autor"e o objeto "João Silva". A Figura 3.1 ilustra um exemplo de grafo RDF, em que os nós representam, da esquerda para a direita, o sujeito e o objeto, enquanto que a aresta representa o predicado.

Além disso, para adicionar semântica para as máquinas, é possível utilizar o *Resource Description Framework in Attributes* (RDFa). Com o RDFa, é possível adicionar marcadores legíveis por máquinas aos atributos HTML, especificando o significado e a estrutura

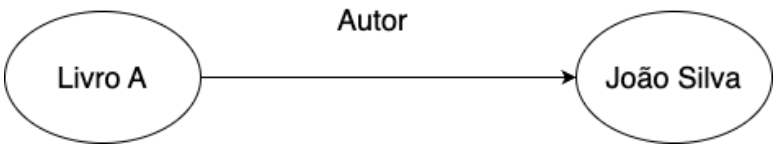

Figura 3.1: Exemplo de grafo RDF

dos dados disponibilizados na página (HERMAN et al., 2015). Dessa forma, as máquinas podem interpretar e processar as informações de forma mais precisa e automatizada.

SPARQL

SPARQL (*SPARQL Protocol and RDF Query Language*) é uma linguagem de consulta semântica para dados estruturados usando o padrão RDF, permitindo buscar e extrair informações específicas com base em padrões, relações e propriedades (GROUP, 2013). Com essa linguagem, é possível fazer pesquisas em um *endpoint* SPARQL, que é um servidor que hospeda um ou mais conjuntos de dados RDF, por meio do protocolo HTTP (FEIGENBAUM et al., 2013). A documentação fornecida pelo W3C sobre a linguagem é dividida em diversos tópicos que abordam suas funcionalidades.

Assim como outras *Query Languages*, o SPARQL possui a capacidade de filtrar, ordenar e limitar resultados com base em condições lógicas, além de oferecer suporte a funções e operadores agregados para a manipulação de dados, como *COUNT*, *SUM*, *AVG*, *MIN* e *MAX*.

A partir da palavra-chave `SERVICE`, é possível mesclar dados distribuídos na *Web* para solicitar informações externas e obter respostas (HARRIS; SEABORNE; GROUP, 2013). Também é possível realizar operações em grafos RDF, como criação, atualização e remoção, utilizando uma sintaxe derivada da linguagem de consulta para RDF.

O SPARQL desempenha um papel essencial na *Web* Semântica, pois possibilita a extração de informações significativas de dados interconectados. Além disso, permite busca precisa, extração de conhecimento estruturado e descoberta de padrões e relações entre os dados interconectados.

A seguir, são apresentados exemplos de consultas SPARQL que demonstram sua aplicação prática:

Listing 3.1: Recuperar todos os recursos de um tipo específico:

```
SELECT ?resource
WHERE {
  ?resource rdf:type
    <http://exemplo.com/ontologia#TipoDeRecurso>.
}
```

Neste exemplo, a consulta recupera todos os recursos que possuem o tipo

```
<http://exemplo.com/ontologia\#TipoDeRecurso>.
```

Listing 3.2: Buscar informações específicas de um recurso

```
SELECT ?nome ?dataNascimento
WHERE {
  <http://exemplo.com/recursos/Pessoa123> foaf:name ?nome;
   foaf:birthday ?dataNascimento.
}
```

Nesta consulta, são recuperados o nome e a data de nascimento do recurso

```
<http://exemplo.com/recursos/Pessoa123>
```

por meio das propriedades `foaf:name` e `foaf:birthday` da ontologia FOAF.

Listing 3.3: Consulta com filtro baseado em propriedades

```
SELECT ?nome
WHERE {
  ?pessoa foaf:name ?nome;
          foaf:age ?idade.
  FILTER (?idade >= 18)
}
```

Neste exemplo, são recuperados os nomes das pessoas com idade igual ou superior a 18 anos, utilizando a propriedade `foaf:age`.

Listing 3.4: Consulta com ordenação dos resultados:

```
SELECT ?nome ?dataNascimento
WHERE {
  ?pessoa foaf:name ?nome;
          foaf:birthday ?dataNascimento.
}
ORDER BY ?dataNascimento
```

Nesta consulta, são recuperados os nomes e as datas de nascimento das pessoas, ordenados em ordem crescente com base na propriedade `foaf:birthday`.

Esses exemplos ilustram algumas das possibilidades de consultas com o SPARQL, demonstrando como é possível buscar informações específicas, filtrar resultados e ordená-los.

Turtle

Turtle (Terse RDF Triple Language), é uma linguagem capaz de descrever um grafo RDF de forma textual (PRUD'HOMMEAUX et al., 2014). A gramática do *Turtle* para triplas é um subconjunto do SPARQL para triplas. Ela oferece acesso a diversos conceitos para descrever um recurso. Os elementos que compõem a linguagem são Triplas Simples, Listas de Predicados, Lista de Objetos, IRIs, Literais, Nós Anônimos, Coleções e Triplas Citadas. Boa parte dos arquivos de configuração dos Solid Pods são escritos em Turtle.

Triplas Simples permitem a escrita de uma tripla RDF com a mesma estrutura usada nos grafos: Sujeito, Predicado e Objeto.

Listing 3.5: Exemplo de Tripla Simples com uso de Listas de Predicados

```
@prefix ex: <http://example.org/> .

ex:John ex:age "30"^^xsd:integer ;
        ex:hasFriend ex:Mary .

ex:Mary ex:age "25"^^xsd:integer ;
        ex:hasFriend ex:John .
```

Nesse exemplo, define-se tanto a idade de John e Mary quanto define-se que eles são amigos. Note que cada componente do grafo é separado por um espaço.

O exemplo acima também inclui Listas de Predicados: ao mesmo tempo que é definida a idade do John por meio do predicado `ex:age`, também é definida a relação de amizade com Mary por meio do predicado `ex:hasFriend`. O mesmo é feito para Mary. É uma forma de açúcar sintático para escrever triplas.

A Lista de Objetos é outro tipo de açúcar sintático, mas voltado para o reaproveitamento do Sujeito e Predicado por Objetos. Os Objetos são separados por ','.

Listing 3.6: Exemplo de Lista de Objetos

```
@prefix ex: <http://example.org/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

ex:book1 dc:title "Harry Potter and the Sorcerer's Stone",
           "Harry Potter e a Pedra Filosofal" ;
         dc:author "J.K. Rowling" ;
         dc:year "1997" .
```

Um *Internationalized Resource Identifier* é delimitado por meio dos caracteres '<' e '>' e podem ser escritos de forma absoluta ou relativa. Uma IRI relativa no formato

<#example1>, é resolvida de acordo com uma IRI base definida por meio da diretiva '@base' ou 'BASE', ou seja, caso a IRI base seja <http://example.com> a IRI relativa será resolvida como <http://example.com/#example1>. Os prefixos podem ser usados como representações de IRIs e podem ser definidos por meio da diretiva '@prefix' ou 'PREFIX'. Além disso, a linguagem define que o token 'a' é resolvido para a IRI

```
http://www.w3.org/1999/02/22-rdf-syntax-ns\#type
```

Listing 3.7: Exemplo com uso do token 'a' e das diretivas '@prefix' e '@base'

```
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .

@base <http://example.com/products/> .

ex:Product1 a schema:Product ;
    schema:name "Smartphone" ;
    schema:brand ex:Brand1 ;
    schema:price "500" .

ex:Brand1 a schema:Brand ;
    schema:name "TechCo" ;
    schema:website <#website> .

<#website> a schema:webSite ;
    schema:url "https://example.com/products/techco" ;
    schema:description "Visit our official website" .
```

O exemplo acima também mostra como são representados os Literais, mais especificamente, no caso das *strings*. Valores e datas também podem ser escritos como Literais. Eles também podem ser tipados, como no exemplo 3.5 por meio de diretivas como ^^xsd:string, ^^xsd:decimal e ^^xsd:boolean (PRUD'HOMMEAUX et al., 2014).

Nós anônimos podem ser representados em Turtle por meio da expressão '_:', seguido de uma série de caracteres de nome. Eles são úteis para criar estruturas sem ter que nomear todos os recursos ou para recursos sem IRI específica. Também é possível criar Nós anônimos com o uso da expressão '[ ]'.

Listing 3.8: Exemplo de uso de Nós anônimos (Blank nodes)

```
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
```

```
ex:Library ex:contains [
    schema:bookTitle "Book 1" ;
    schema:author [
        schema:name "Author 1"
    ]
] ;
ex:contains [
    schema:bookTitle "Book 2" ;
    schema:author _:author2
] .

_:author2 a schema:Person ;
    schema:name "Author 2" .
```

As Coleções permitem criar listas em RDF, o que também ajuda na montagem de estruturas. Para isso é usada a expressão '(...)', em que '...' pode ser nada ou um termo RDF. Coleções devem ser usadas como Sujeito ou Objeto e também são consideradas Nós anônimos.

Listing 3.9: Exemplo de uso de uma Coleção

```
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .

ex:Book1 a schema:Book ;
    schema:title "The Great Gatsby" ;
    schema:authors (
        ex:Author1
        ex:Author2
        ex:Author3
    ) .

ex:Author1 a schema:Person ;
    schema:name "F. Scott Fitzgerald" .

ex:Author2 a schema:Person ;
    schema:name "Ernest Hemingway" .
```

```
ex:Author3 a schema:Person ;
    schema:name "Virginia Woolf" .
```

Triplas Citadas (*Quoted Triples*) podem ser representadas ao colocar uma tripla precedida e sucedida pelas expressões '«' e '»', respectivamente. Elas são utilizadas no formato *Turtle* e em outros formatos RDF para agrupar informações relacionadas em uma única estrutura. Elas têm o objetivo de facilitar a leitura e compreensão do grafo RDF, além de preservar o contexto e reduzir a redundância. As quoted triples permitem criar estruturas complexas e representar relações aninhadas, contribuindo para a organização eficiente dos dados RDF. Sua utilização promove a expressividade e a compactação dos dados, melhorando a representação de informações relacionadas em um grafo RDF.

Linked Data Platform

A Plataformas de Dados Ligados (*Linked Data Platform* — LDP) é uma especificação da W3C que busca padronizar a forma de acessar dados da *Web*, a fim que que possa ser acessados tanto por máquinas tanto por pessoas. Essa especificação é baseada em quatro regras:

- Usar URIs como nome de coisas;
- Usar URIs de HTTP para que pessoas possam procurar esses nomes;
- Quando alguém procura uma URI, é providenciado informações uteis usando os padrões SPARQL e RDF;
- Incluir links para outras URIs, para que essas descubram mais coisas.

3.2.2 Tecnologias para introdução pedagógica ao Solid na graduação em computação

Typescript

O TypeScript[1] é uma linguagem de programação de código aberto desenvolvida pela Microsoft, baseada no JavaScript, mas com recursos adicionais, como tipos estáticos, interfaces, módulos e decorators. A principal proposta da linguagem é não alterar a forma como o JavaScript é executado, atuando apenas no controle de tipos para evitar erros e falhas no código.

O JavaScript é uma linguagem interpretada que não possui tipagem estática por padrão, sendo totalmente baseada em tipagem dinâmica. Isso significa que o programador

[1] <https://www.typescriptlang.org/>

não precisa declarar explicitamente o tipo das variáveis, pois a linguagem infere automaticamente o tipo com base no contexto. No entanto, essa abordagem traz a possibilidade de ocorrência de *bugs*, pois diferentes tipos de dados podem ou não possuir um método ou aceitar uma determinada operação em comum. Por exemplo, o operador "+"pode ser utilizado tanto para concatenar strings quanto para somar números, resultando em comportamentos diferentes: 2 + 2 resulta em 4, enquanto '2' + '2' resulta em '22'.

Ao ser compilado, o TypeScript é convertido integralmente em JavaScript, sem deixar vestígios da tipagem utilizada no código-fonte. Isso significa que o código executará da mesma forma que foi escrito, mesmo que haja erros de tipo. Além disso, o TypeScript permite uma transição suave para o JavaScript, não exigindo o uso de um framework específico.

Essa tecnologia possibilita a detecção de erros de sintaxe e lógica em tempo de compilação, o que aumenta a qualidade e a segurança do código, além de melhorar a produtividade do desenvolvedor. A linguagem também oferece suporte a ferramentas de desenvolvimento, como editores de código, depuradores e testes automatizados.

Por fim, o TypeScript facilita a integração com diversas bibliotecas e *frameworks* populares, permitindo a instalação de definições de tipos para essas bibliotecas. Dessa forma, é possível utilizar o TypeScript em conjunto com *frameworks* como React, Angular e Vue.js, por exemplo. Com o TypeScript, o código fica mais legível, limpo e reutilizável, evitando erros comuns e problemas relacionados à tipagem dinâmica do JavaScript.

ReactJS

O ReactJS, também conhecido como React, é uma biblioteca JavaScript desenvolvida por Jordan Walke, ex-funcionário do Facebook. Sua primeira implementação ocorreu em 2011 e posteriormente foi disponibilizado como código aberto na JSConf US, uma conferência da comunidade JavaScript realizada nos Estados Unidos, em maio de 2013. A criação do React teve como objetivo facilitar a manutenção do Facebook, que enfrentava desafios devido ao tamanho extenso de seu código e ao envolvimento de múltiplos engenheiros, o que afetava o desempenho da página.

O React é especialmente adequado para a criação de *Single Page Applications* (SPAs), que buscam proporcionar aos usuários uma experiência de alta velocidade e fluidez. Para isso, o React utiliza uma abordagem em que a estrutura da página *Web* é construída e suas partes são atualizadas conforme o usuário interage com elas, evitando a necessidade de recarregar toda a página a cada interação. Essa funcionalidade é implementada por meio de um algoritmo de Virtual DOM (Modelo de Documento por Objetos). O DOM é uma convenção universal para representar objetos usando HTML ou XML. Ele representa os documentos como uma árvore de nós, na qual cada nó pode ter filhos e alguns são sempre

folhas da árvore. As especificações do DOM são definidas no documento da W3C (*World Wide Web Consortium*), uma organização criada por Tim Berners-Lee para padronizar a *Web*, e incluem a estrutura do DOM e algoritmos relacionados a mutações na árvore. O algoritmo de Virtual DOM cria uma cópia do DOM real na memória e realiza comparações para atualizar apenas o necessário.

O React adota a abordagem de dividir a interface em componentes, que são blocos de código reutilizáveis capazes de receber dados e renderizar elementos na tela. Além disso, oferece um sistema de gerenciamento de estados que permite armazenar e atualizar dados de forma consistente e eficiente. Para facilitar a escrita do código, o React utiliza uma sintaxe chamada JSX (ou TSX quando utilizado com TypeScript), que possibilita uma mistura de HTML e JavaScript.

É importante ressaltar que a manutenção do React não está limitada apenas aos funcionários da Meta (antigo Facebook), mas também envolve diversas empresas e desenvolvedores independentes. Essa ampla comunidade de colaboradores contribui para uma extensa documentação, o que facilita a resolução de problemas e a curva de aprendizado relacionada ao React.

Tailwind CSS

O Tailwind CSS é um *framework* de código aberto amplamente utilizado que simplifica a criação de designs personalizados e responsivos. Criado por Adam Wathan, Jonathan Reinink, David Hemphill e Steve Schoger, ele se destaca por permitir a escrita direta de classes no HTML, o que proporciona uma maior produtividade para aqueles familiarizados com esse estilo de codificação.

Diferentemente de outros *frameworks* CSS que oferecem classes pré-definidas para componentes completos, o Tailwind CSS adota uma abordagem baseada em classes utilitárias, que podem ser combinadas para estilizar componentes de forma modular. Além disso, o *framework* oferece flexibilidade, permitindo personalização e a adição de classes e *plugins* conforme as necessidades específicas de cada projeto.

Uma das principais características do Tailwind CSS é a sua abordagem "mobile-first", que prioriza a construção da interface para dispositivos móveis antes de abordar a versão *desktop*. O *framework* fornece prefixos de *breakpoints* bem definidos para diferentes tamanhos de dispositivos, correspondendo às media *queries* comumente usadas em CSS convencional. Esses prefixos permitem que a interface se adapte de forma responsiva a diferentes tamanhos de tela. Além disso, o Tailwind CSS também oferece prefixos para estados do componente, como "active" (quando um link está ativo) e "hover" (quando o cursor do mouse está sobre o componente), proporcionando maior controle sobre o comportamento visual.

Em termos de manutenção, o Tailwind CSS conta com uma comunidade ativa e uma documentação completa, o que facilita o aprendizado e oferece suporte aos usuários. Além disso, o *framework* é compatível com ferramentas modernas de desenvolvimento *Web*, incluindo pré-processadores, *bundlers* e *frameworks* JavaScript, permitindo uma integração suave em fluxos de trabalho existentes.

Em resumo, o Tailwind CSS é uma poderosa ferramenta para o desenvolvimento de interfaces personalizadas e responsivas. Sua abordagem baseada em classes utilitárias, combinada com a flexibilidade de personalização e a ampla compatibilidade, torna-o uma escolha popular entre os desenvolvedores que buscam eficiência e controle no processo de criação de designs *Web*.

Next.js

O Next.js, ou Next, é um *framework fullstack* de React para o desenvolvimento de aplicações *Web*. Seu propósito é fornecer um conjunto de ferramentas e configurações que permitem que o usuário foque no desenvolvimento da aplicação, sem ter que ficar configurar as ferramentas. O Next é mantido pela Vercel, que também fornece uma plataforma para hospedar projetos, com integração com o GitHub de forma que updates na branch main são enviados para produção, permitindo uma integração constante (*Constant Integration*).

O Next estabelece uma arquitetura pré-definida para o projeto, de forma que o roteamento é integrado de acordo com a estrutura de pastas do projeto. Na versão atual (versão 13), o Next permite que as rotas sejam definidas por meio de pastas contendo um arquivo *page.tsx*. Por exemplo, estando a página principal no URL *www.exemplo.com*, a rota *www.exemplo.com/texto* dessa página pode ser definida por meio da criação de um diretório "texto" na pasta "app" do projeto, contendo um arquivo "page.tsx".

Além da convenção da nomenclatura "page.tsx", o *framework* possui outras convenções para componentes da aplicação que permitem incrementar cada rota com interfaces para casos de erro, interface de carregamento, *endpoint* de API e interface padrão para rotas paralelas. São elas: a nomenclatura *layout.tsx*, usada para definir uma UI (User Interface) comum para páginas; o arquivo de nome *loading.tsx*, para definir uma UI utilizada para indicar o carregamento da página ou de um componente pro usuário; o arquivo *not-found.tsx*, para definir uma UI a ser mostrada quando uma rota não for encontrada; *error.tsx*, para definir uma UI a ser mostrada quando ocorrem erros em rotas aninhadas; *global-error.tsx*, uma UI a ser mostrada quando houver um erro que não for lidado por nenhum arquivo *error.tsx*; *route.ts* para lidar com requisições de forma customizada por meio da API Request/Response; *template.tsx*, semelhante ao *layout.tsx*, porém não persiste entre rotas e é remontado a cada navegação, os elementos são recriados na DOM,

os estados não são preservados e os efeitos são ressincronizados. No entanto, para a confecção do material pedagógico foram utilizadas apenas o *layout.tsx*, e o *page.tsx*, por se tratar de uma página estática.

Outras características ofertadas pelo Next são a renderização no lado do cliente e do servidor com componentes para cada um deles, renderização estática e dinâmica otimizada no servidor, *streaming* na borda em ambientes Node.js, suporte para métodos de estilização como Tailwind CSS, otimizações de imagem e fonte e suporte para Typescript.

Next.js é usado por grandes empresas da *Web*, como Netlix, Uber, Airbnb e Github e permite a criação de aplicações *Web* interativas, dinâmicas e rápidas.

Vercel

O presente trabalho empregou o Vercel[2] como plataforma de implantação para o *deploy* de material no formato *Web*. O Vercel, uma plataforma de nuvem sem servidor, foi selecionado devido à sua capacidade de hospedar sites e serviços online que são lançados instantaneamente, escaláveis e dispensam a necessidade de monitoramento extensivo, requerendo configurações mínimas ou nulas.

Esta ferramenta foi escolhida por oferecer um processo de *deploy* ágil e simplificado. Além disso, é compatível com diversos *frameworks*, incluindo os renomados Angular, Gatsby, React, Vue, Next e Nuxt. Vale ressaltar que o Vercel é desenvolvido pelos criadores do Next.js, um *framework* utilizado no desenvolvimento do tutorial, permitindo uma integração eficiente entre o *framework* e a plataforma, simplificando o processo de *deploy* da aplicação com suporte a *CI/CD* (*Continuous Integration/Continuous Deployment*).

O Vercel disponibiliza soluções abrangentes em termos de infraestrutura, armazenamento, fluxo de trabalho colaborativo, segurança, bem como ferramentas para o *deploy*, como integração com GitHub, GitLab, BitBucket e Vercel CLI. Além disso, a plataforma oferece recursos de monitoramento, detalhamento e *logs* da implantação, análise de desempenho (*Web Analytics*), métricas de velocidade (*Speed Insights*) e validação de dados *Open Graph*[3]. Também é possível visualizar cada *branch* durante o processo de implantação.

Importante ressaltar que nem todos os recursos estão disponíveis em todos os planos de uso. Considerando que o projeto do presente estudo é de cunho educacional e não visa lucratividade, optou-se pelo plano gratuito denominado *Hobby*. Esse plano oferece funcionalidades de *deploy* por meio de linha de comando ou integração com Git, *CI/CD* incorporado, HTTPS/SSL automático e prévias para *pushes* no Git. Dentre as funcionalidades fornecidas pelo plano gratuito, destaca-se a facilidade de *deploy* em virtude da

[2] <https://vercel.com/>
[3] <https://ogp.me/>

integração com o Git, sendo este o aspecto mais atrativo para a realização do *deploy* do material em questão e o que definiu a escolha da ferramenta em questão.

Capítulo 4

Implantação de um provedor Solid-POD para ecossistema educacional

Este capítulo apresenta a utilidade do projeto e os passos necessários obtidos com esse estudo, sendo dividido em três seções: O provedor na interface pretendida; Estudo preliminar e implantação em *localhost*. A primeira seção descreve uma proposta de utilização do solid Pod em um ambiente universitário, bem como suas limitação e ações necessárias. Já a segunda seção descreve o passo a passo dos teste, bem como o empecilhos encontrados durante esses testes.

4.1 O Provedor na Interface Pretendida

A interface que está sendo desenvolvida para o ecossistema educacional será resultado do Trabalho de Conclusão de Curso (TCC) (JUVITO; SOARES, 2023) e segue a arquitetura do departamento de Ciência da Computação da UnB. Essa interface proporcionará um ambiente virtual que replica o departamento, oferecendo maneiras de acessar os diversos componentes do ecossistema. Serão utilizados recursos inspirados em plataformas de metaverso, que incluem objetos interativos para acesso a conteúdos incorporados, como sites, vídeos e imagens. Por exemplo, interagir com um *banner* virtual poderá levar o usuário a um conteúdo disponibilizado na plataforma CiCFriends, uma instância do Friendica.

Durante a criação do Pod, o provedor implantado na UnB será recomendado aos usuários. No entanto, seguindo a abordagem do projeto Solid, os usuários terão a liberdade de escolher qualquer provedor para o seu Pod na aplicação, podendo inclusive movê-lo para outros provedores posteriormente, caso desejem. À medida que os usuários interagem

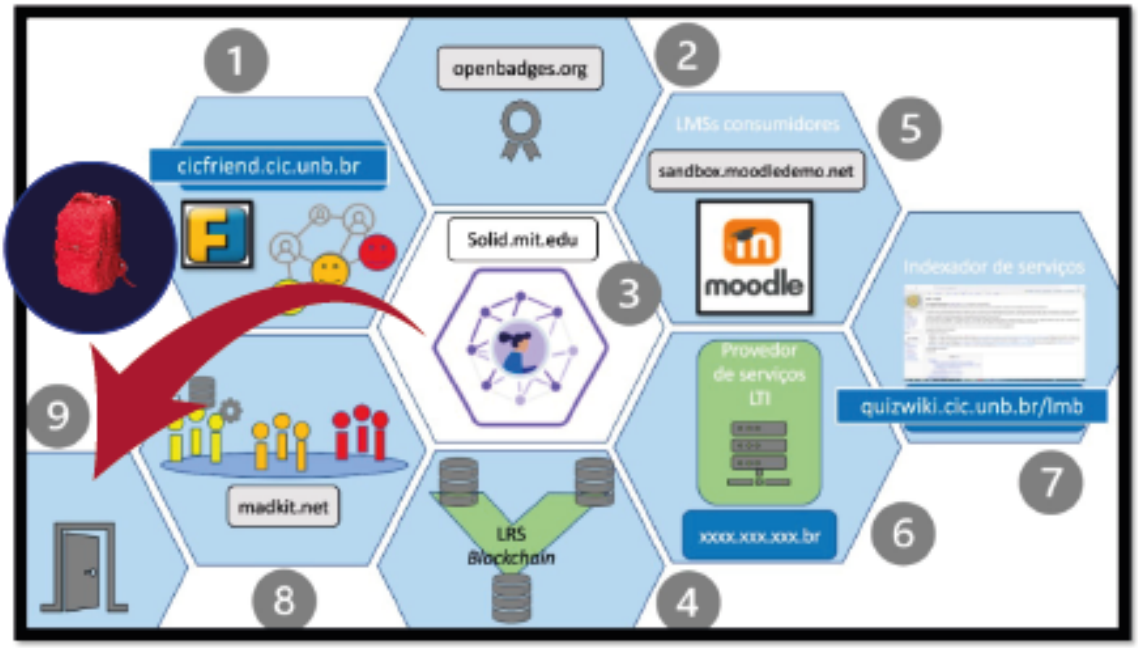

Figura 4.1: O Pod como uma mochila na metaversidade

com a interface, outros componentes do ecossistema serão acionados e poderão interagir com o Pod, desde que o usuário conceda permissão para tal.

A capacidade de movimentar o Pod, armazenar dados e escolher o que pode ser acessado e como, permite usar metáforas como a comparação do Pod a um pendrive pessoal. No contexto educacional, uma metáfora que se encaixa bem é a visão do Pod como uma mochila, já que ela relaciona-se com a capacidade de armazenamento, portabilidade e a liberdade de escolher o que retirar ou guardar dentro dela. Assim, a ideia é que o usuário possa navegar pelos dados salvos no Pod ao interagir com a mochila, que estará sempre ao seu alcance.

Entretanto, dadas as limitações atuais dos *podbrowsers* (navegadores de Solid pods), será necessário desenvolver uma aplicação intuitiva que não apenas ofereça funcionalidades básicas, como adicionar, excluir, renomear e fornecer acesso aos dados, mas também possibilite abrir arquivos nos formatos mais comuns, como .mp3, .mp4, .txt, .png, .jpeg, .pdf, entre outros, além de permitir a edição de alguns desses arquivos, como é o caso dos arquivos .txt, por exemplo. Isso se faz necessário devido à descontinuação, falta de manutenção constante ou limitações de funcionalidades nos *podbrowsers* recomendados pela seção de aplicativos no site do projeto Solid[1], os quais, em sua maioria, são voltados para desenvolvimento.

[1]<https://solidproject.org/apps>

4.2 Estudo Preliminar e disponibilização em *Localhost*

Antes de começar a implantação, de fato, de um provedor Solid foi necessário compreender suas limitações e suas capacidades. Assim, uma vez que são indicados alguns provedores no site do projeto Solid[2], sendo alguns de código aberto, vários testes foram realizados com o intuito de atingir esse fim. Os teste foram feitos com aplicações clientes disponibilizados no site do projeto[3], e com a aplicação [4], para vários provedores com implementações distintas: Enterprise Solid Server (ESS — um produto comercial de código fechado); Node Solid Server (NSS — um produto comunitário de código aberto); e Community Solid Server (CSS — um produto comunitário de código aberto).

A maior parte das aplicações exploradas na fase de testes com provedores em produção tinham como foco o gerenciamento do Pod. Isso se deve à intenção de verificar a usabilidade do Pod de forma intuitiva, assim como avaliar o custo para utilizar e configurar as funcionalidades prometidas pelo projeto. As principais aplicações testadas foram: Inrupt PodBrowser[5](descontinuado e removido do GitHub), SolidOS[6] (presente em diversos provedores em produção), Penny[7] e Solid Filemanager[8]. A princípio foi observado que o Inrupt Podbrowser, possuía uma boa integração com pods do servidor empresarial hospedado pela própria Inrupt, porém foram notadas limitações de uso para pods de outros provedores; não foi possível esclarecer se esse comportamento é intencional ou não, uma vez que o projeto foi descontinuado. Com relação às outras aplicações, foi notado que as aplicações ainda não oferecem controle sobre todas as funcionalidades do pod.

Da observação dos resultados obtidos a partir dos testes iniciais, pode-se concluir que as aplicações possuem limitações a respeito de compatibilidade, ou seja, uma aplicação que é compatível com um provedor não necessariamente será compatível com outro provedor.

Conhecendo o potencial inicial das aplicações e provedores, foi iniciado os estudos em aplicações *localhost* com todas a implementações Solid Pod disponíveis open source: Node Solid Server (NSS); Community Solid Server (CSS); e solid-nextcloud; e PHP Solid Server. Desses o PHP Solid Server e o solid-nextcloud foram descartados devido a dificuldade de de conseguir testar o servidor, em outras palavras, foi dedicado tempo realizar os testes em *localhost*, porém devido a inúmeros erros, não foi possível testá-los; também vale ressaltar que o PHP Solid Server não recebe atualizações desde o início de 2022. Mas com o Node

[2]<https://solidproject.org/users/get-a-pod#get-a-pod-from-a-pod-provider>
[3]<https://solidproject.org/apps>
[4]<https://pod-chat.com/>
[5]<https://podbrowser.inrupt.com/login?returnTo=%2F>
[6]<https://github.com/SolidOS/solidos>
[7]<https://penny.vincenttunru.com/>
[8]<https://github.com/Otto-AA/solid-filemanager>

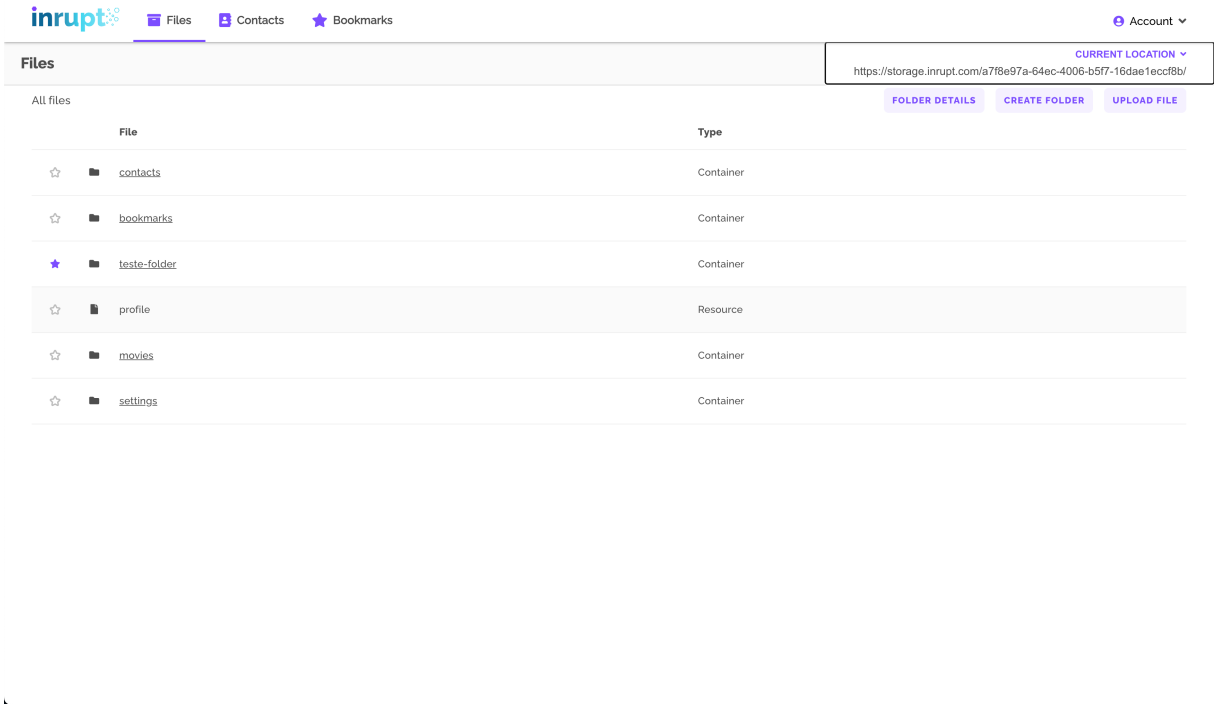

Figura 4.2: Interface da aplicação Inrupt PodBrowser. O URI mostrado na direita indica o endereço do armazenamento.

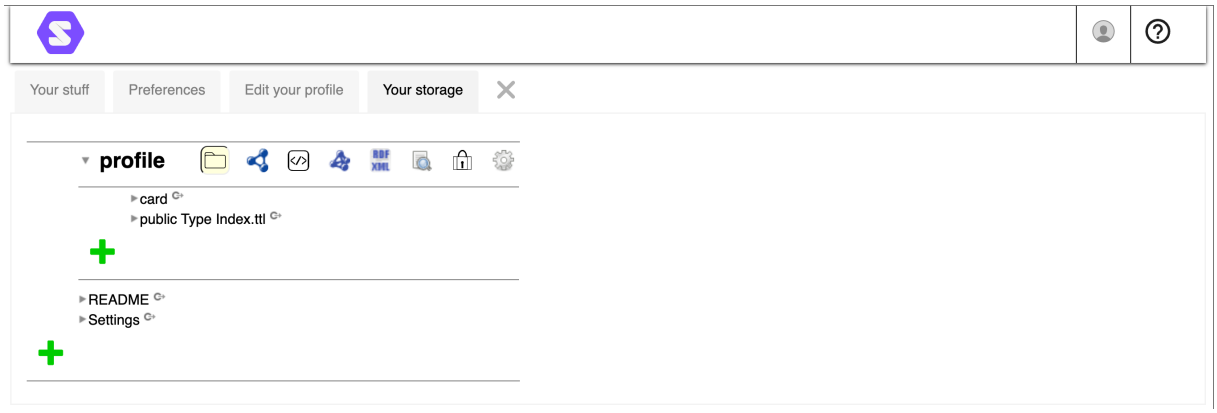

Figura 4.3: Enter Caption

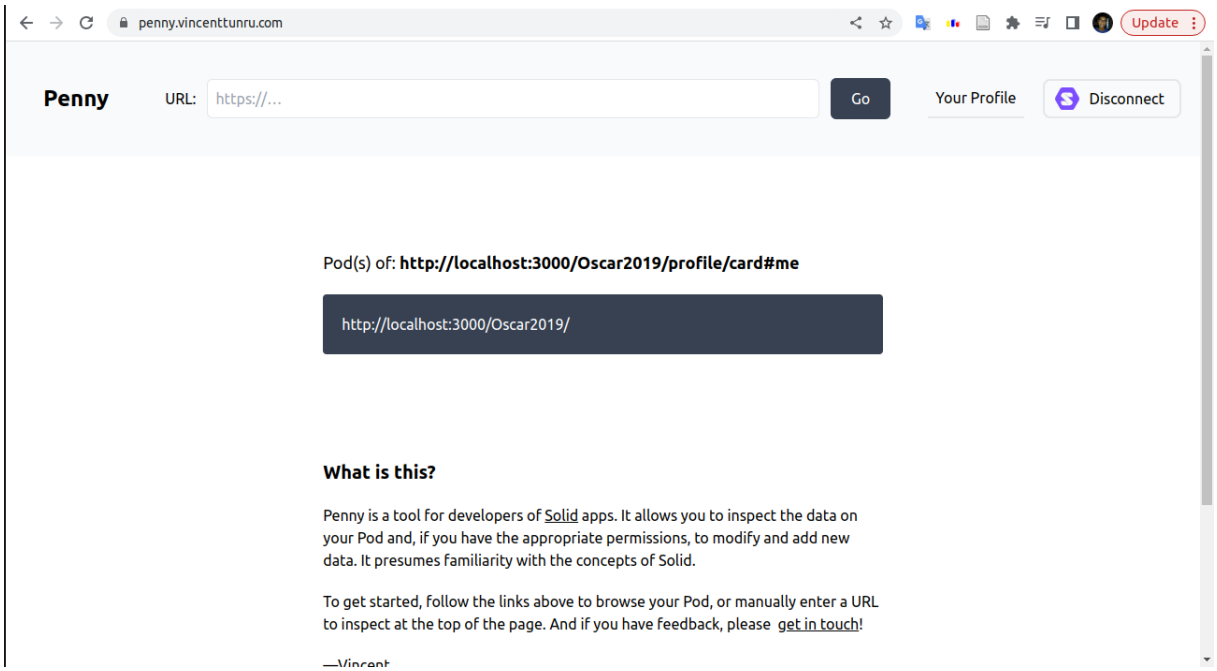

Figura 4.4: Página inicial do Penny quando está conectado a um pod.

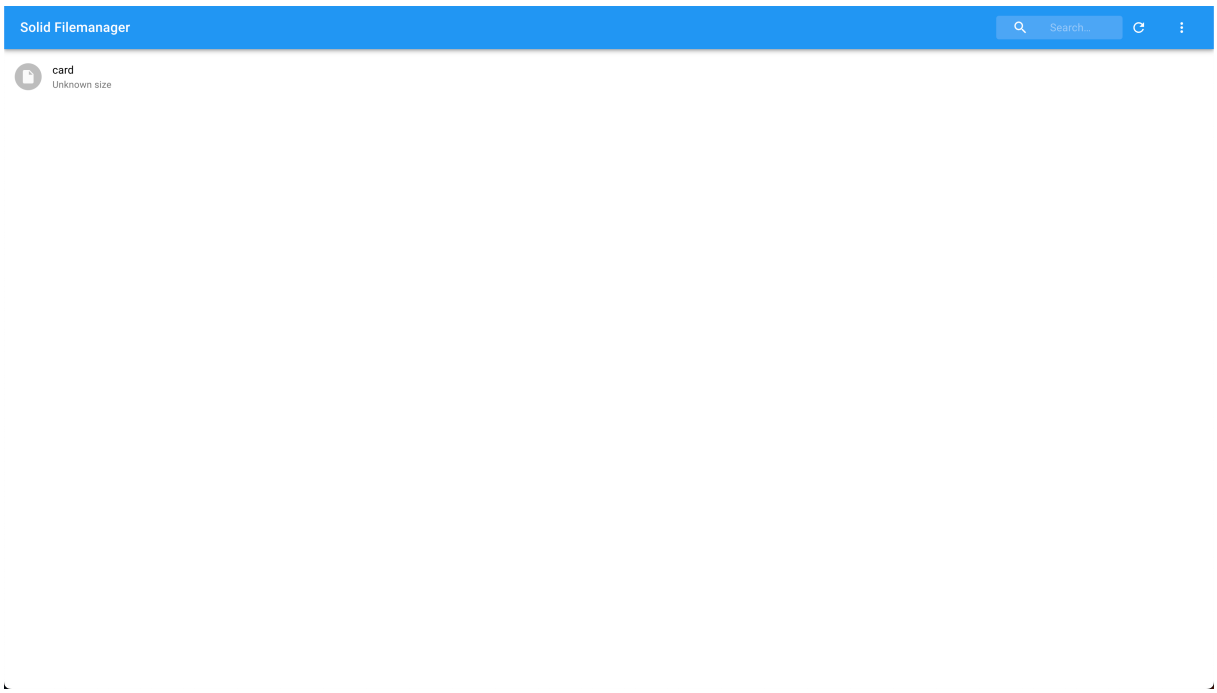

Figura 4.5: Interface do Solid Filemanager.

Solid Server e o Community Solid Server, foi possível completar as instruções de instalação em ambos tipos de provedores, contudo, para o Node Solid Server, apesar das inúmeras tentativas não foi obtido sucesso em fazê-lo funcionar com outras aplicações. Assim foi concluído que a melhor alternativa seria usar o Community Solid Server.

Para testar o Community Solid Server foi seguido o tutorial da página do github[9] com pequenas modificações 4.1, onde foi obtido sucesso para a execução no penny [10]. Entretanto, para outras aplicações apenas o servidor localhost com um certificado de segurança sem assinatura não bastava para seu funcionamento, em virtude desse fato, um dos autores deste trabalho utilizou de seu domínio pessoal pra poder testar de forma mais fidedigna o Community Solid Server. Para adquirir os certificados foram utilizados os serviços *let's encrypt*[11]e *certbot*[12] seguindo os passo em 4.3. Uma vez que o certificado foi obtido, passos semelhante foram foram seguido para o teste em localhost4.2, contudo foi necessário ter algumas alterações na configuração do projeto no Anexo A, além de criar um servidor proxy reverso Fazer o tutorial do proxy reverso.

Listing 4.1: bash version

```
git clone https://github.com/CommunitySolidServer/
   ↪ CommunitySolidServer.git
cd CommunitySolidServer
npm ci
npm start -- -c config/file.json -f data/
```

Listing 4.2: bash version

```
git clone https://github.com/CommunitySolidServer/
   ↪ CommunitySolidServer.git
cd CommunitySolidServer
npm ci
npm start -- -c config.json -f data/ -b https://<Nome do Dominio
   ↪ >/ --httpsKey /etc/letsencrypt/live/<Nome do Dominio>/
   ↪ privkey.pem --httpsCert /etc/letsencrypt/live/<Nome do
   ↪ Dominio>/fullchain.pem
```

Listing 4.3: bash version

```
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

[9] <https://github.com/CommunitySolidServer/CommunitySolidServer>
[10] <https://penny.vincenttunru.com/>
[11] <https://letsencrypt.org/>
[12] <https://certbot.eff.org/>

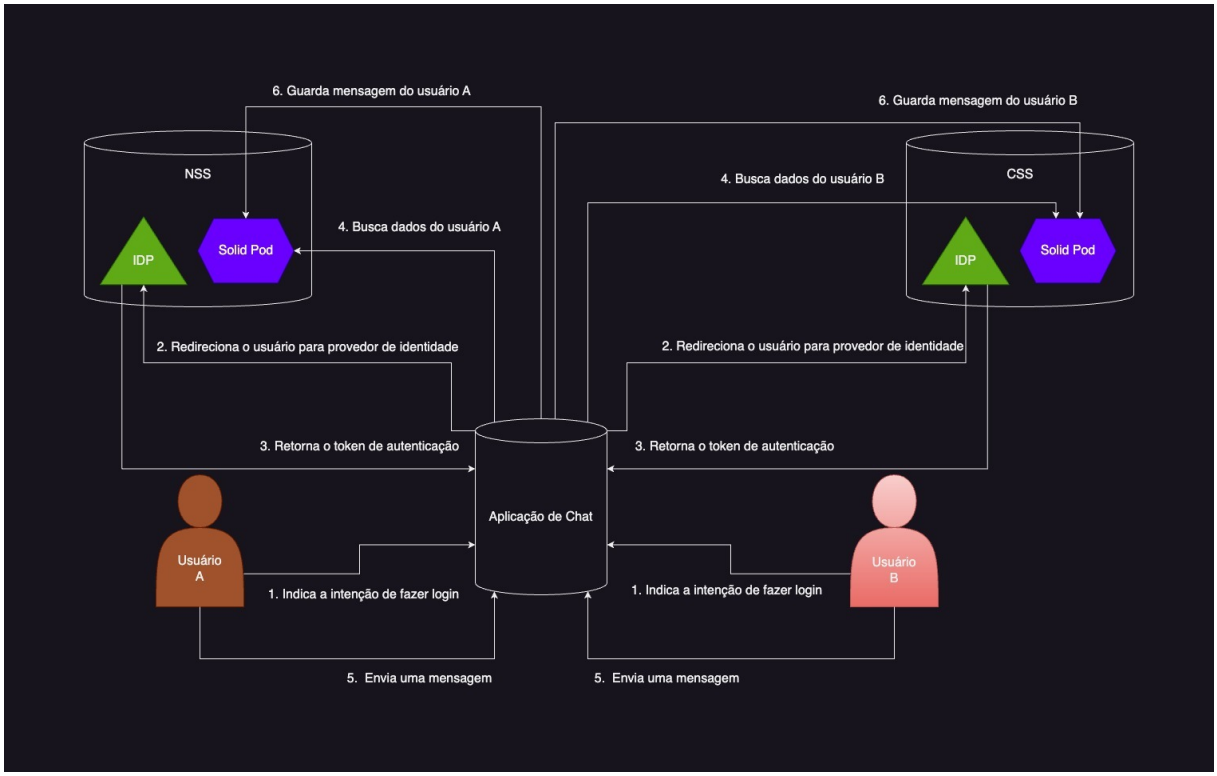

Figura 4.6: Exemplo de funcionamento do chat-pod

```
sudo certbot certonly --standalone
```

Com o servidor implementado foram iniciados os teste de aplicações foco, as aplicações de mensagem de texto. As aplicações de mensagens de texto são as aplicações mais simples que é necessário a interação de duas pessoas, pois, é obviamente uma conversa entre duas pessoas e cada usuário a conversa é tratado como leitor e e escritor, assim não foi pesquisada nenhuma aplicação de *chat* de grupo. E por esse motivo esse tipo de aplicação foi escolhida para compreender a atual capacidade do protocolo Solid de manter a descentralização dos dados do usuário, assim como entender quem tem o poder de criação do código. As aplicações escolhidas que escolhemos para comparar foram o pod-chat[13] e liqid.chat[14]. Com os teste da aplicações foi percebido que a aplicação mais adequada é o chat-pod, pois enquanto o liqid.chat armazena os dados no usuário que criou o *chat*, o chat-pod permite que cada usuário armazenem suas próprias mensagens, exemplo do funcionamento na figura 4.6.

Não esquecendo existe uma adendo que faz com que o pod-chat não funcione no Community Solid Server e que funcione no Node Solid Server. O método de investigação foi fazer o *debug* linha a linha do código, onde foi percebido que o pod-chat busca informações

[13]<https://pod-chat.com/>
[14]<https://liqid.chat/>

dentro do *card* do usuário, e em outros locais, e como solução para esse problema basta colocar as informações adicionais dentro do pod:

- Uma pasta nomeada de *inbox*, com permissão de escrita e de anexar para todos os usuários;
- Uma pasta nomeada settings, com dois arquivos: privateTypeIndex.ttl; e publicTypeIndex.ttl;
- Adicionar *trustedApp*, *storage*, *account*, *privateTypeIndex*, *publicTypeIndex*, *name*

E mesmo colocando informações adicionais, ainda percebe-se um funcionamento abaixo do esperado ao utilizar o pod-chat juntamente com o CSS. Assim mostrando de forma mais clara que o protocolo Pod ainda está em em uma fase imatura.

Também a partir do experimento foi possível obter uma melhor compreensão sobre os arquivos turtle, os arquivos de lista de controle de acesso, bem como os arquivos criados pelo pod-chat. Parte do processo pode ser percebido através das figuras 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18 e 4.19.

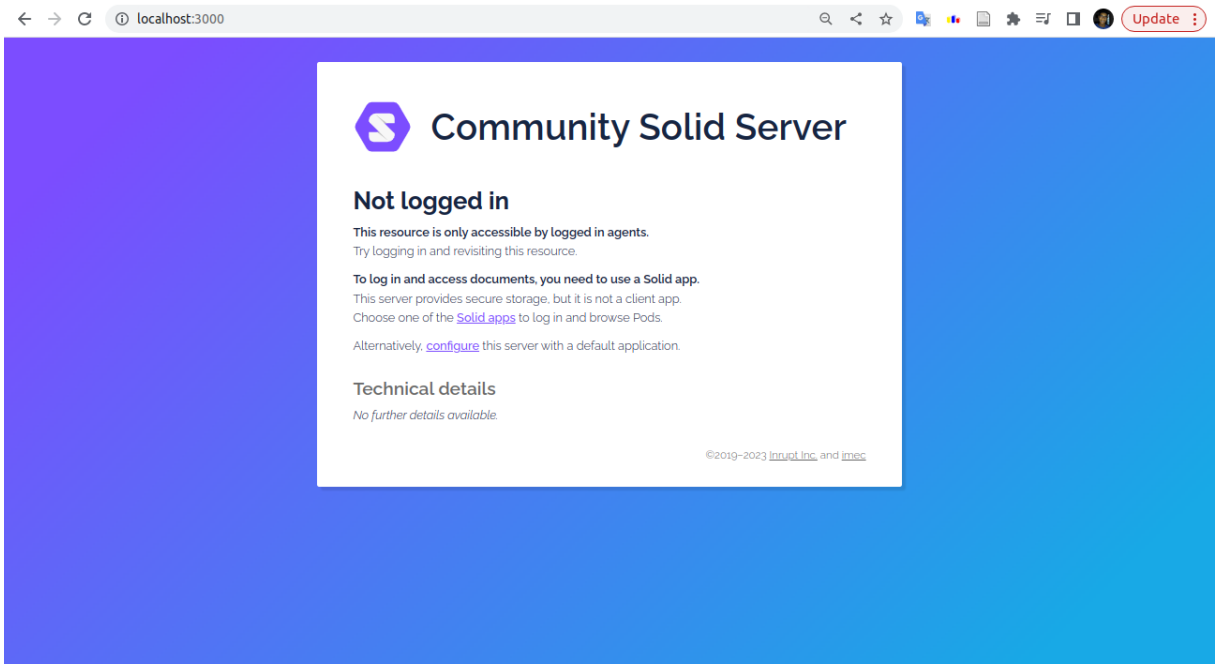

Figura 4.7: Imagem do CSS sem realiza o login.

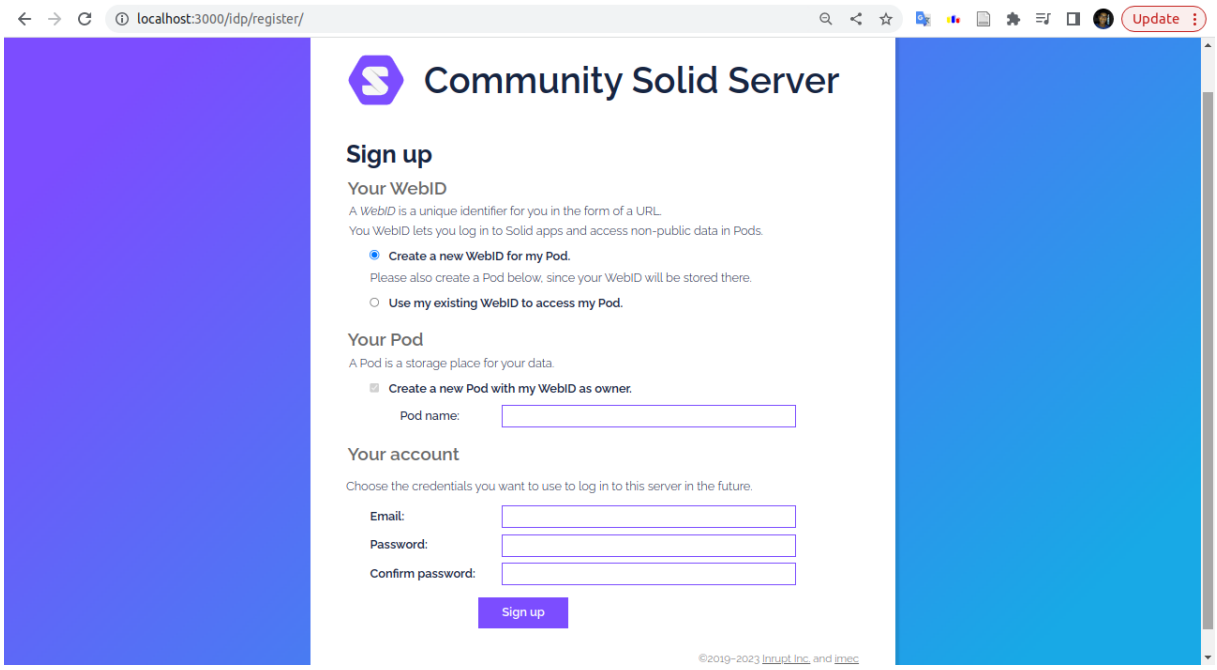

Figura 4.8: Página de registro.

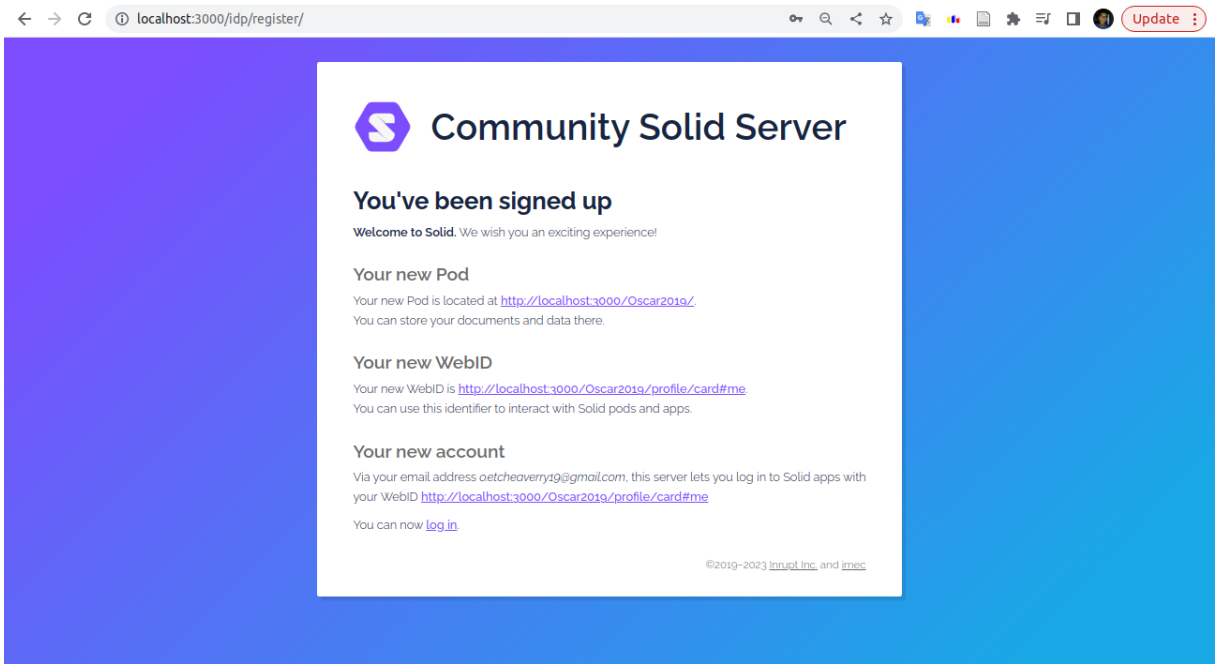

Figura 4.9: Página de registro concluído.

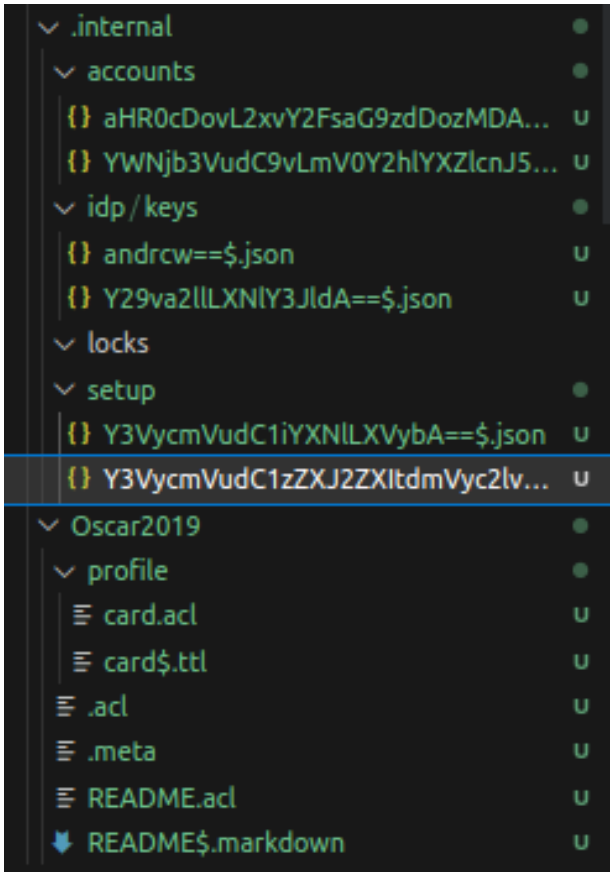

Figura 4.10: Arquivos criado pelo servidor.

Figura 4.11: Arquivos com os dados da conta

```
# ACL resource for the WebID profile document
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

# The WebID profile is readable by the public.
# This is required for discovery and verification,
# e.g. when checking identity providers.
<#public>
    a acl:Authorization;
    acl:agentClass foaf:Agent;
    acl:accessTo <./card>;
    acl:mode acl:Read.

# The owner has full access to the profile
<#owner>
    a acl:Authorization;
    acl:agent <http://localhost:3000/Oscar2019/profile/card#me>;
    acl:accessTo <./card>;
    acl:mode acl:Read, acl:Write, acl:Control.

```

Figura 4.12: Arquivo de autorização de acesso ao card.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix solid: <http://www.w3.org/ns/solid/terms#>.

<>
    a foaf:PersonalProfileDocument;
    foaf:maker <http://localhost:3000/Oscar2019/profile/card#me>;
    foaf:primaryTopic <http://localhost:3000/Oscar2019/profile/card#me>.

<http://localhost:3000/Oscar2019/profile/card#me>

    solid:oidcIssuer <http://localhost:3000/>;
    a foaf:Person.
```

Figura 4.13: Arquivo com a descrição do card.

```
# Root ACL resource for the agent account
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

# The homepage is readable by the public
<#public>
    a acl:Authorization;
    acl:agentClass foaf:Agent;
    acl:accessTo <./>;
    acl:mode acl:Read.

# The owner has full access to every resource in their pod.
# Other agents have no access rights,
# unless specifically authorized in other .acl resources.
<#owner>
    a acl:Authorization;
    acl:agent <http://localhost:3000/Oscar2019/profile/card#me>;
    # Optional  Follow link (ctrl + click)  used for account recovery:
    acl:agent <mailto:o.etcheaverry19@gmail.com>;
    # Set the access to the root storage folder itself
    acl:accessTo <./>;
    # All resources will inherit this authorization, by default
    acl:default <./>;
    # The owner has all of the access modes allowed
    acl:mode
        acl:Read, acl:Write, acl:Control.
```

Figura 4.14: Arquivo de autorização de acesso da pasta raiz do pod.

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix solid: <http://www.w3.org/ns/solid/terms#>.
@prefix space: <http://www.w3.org/ns/pim/space#>.
@prefix ldp: <http://www.w3.org/ns/ldp#>.

<https://oscarmadureira.com.br/madureira/profile/card> a foaf:PersonalProfileDocument;
    foaf:maker <#me>;
    foaf:primaryTopic <#me>.
<#me> a foaf:Person;
    solid:oidcIssuer <https://oscarmadureira.com.br/>;
    acl:trustedApp _:n3-1;
    ldp:inbox <https://oscarmadureira.com.br/madureira/inbox/>;
    space:storage <https://oscarmadureira.com.br/madureira/>;
    solid:account <https://oscarmadureira.com.br/madureira/>;
    solid:privateTypeIndex <https://oscarmadureira.com.br/madureira/settings/privateTypeIndex.ttl>;
    solid:publicTypeIndex <https://oscarmadureira.com.br/madureira/settings/publicTypeIndex.ttl>;
    foaf:name "Oscar".
_:n3-1 acl:mode acl:Append, acl:Control, acl:Read, acl:Write;
    acl:origin <https://pod-chat.com>.
<https://www.pod-chat.com/RSAPublicKey> <http://rdfs.org/sioc/ns#content_encoded>
"MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA3gq19wZoqYEpvhUNeCJHcZ+pqbKQU0LeLeN0Esgpij1r/mdRh1net/WS4GhTqLigGttoJ
+AqPXZ31ct9PvGKVUvcw1PngEpjZywrGSlFzVzvGhnc/bEPBVTGfrq3f+Z9zLDi1T3Xi53dQpH//yFqJ
+qPSAjEQANPE260IL5jsrrCfbR8VfEZUUnpcWFGSQZzBSjB6rR/XMT/u0s/XUmbSgb2T6f52X08GHnijG9CKGZZ9NVS11FelsSAFYEIv/5nFucKMFF
+k+ADSO5NfF0Gv7GCQxbkXyZWPrxBoY3KkLJ2OHvlc1ra343rqxfwQPmn5uBrEyIm76eSeLfkYLEFpaFf/LXN
+rAP1u4Ui3a24wSLqEbiwpO3FabSXRpnOl5I2ELPoHhoLT52EuYlyNVAWez1H7INdiSN5Kq4gt721B0xDWfoDjbwn9qwewP1Ol4US2DmXPG2loYn9m
KtoXDYx04vfzNLtTBPrTy7fEIFyjjdUbWn2KvN6VdCsvzujWvKg3Oyv3rInlJLSH/Z09nntf0fBa
+clkQrrK6u2mNBD6IYzfTqb08et83aXqWmcC7lHHodW3
+f21fZT2QyBDhfpDOCYy3eWf9javC4wtY4bNgp7gkUXP4yk5uVvmQ8OIc8yv3m2qMqvnIlOxKFIdjzQuB25HE/LIXLYEYFJWZjlS0CAwEAAQ==".

```

Figura 4.15: Exemplo de um card com os dados para tornar o pod-chat.com funcional.

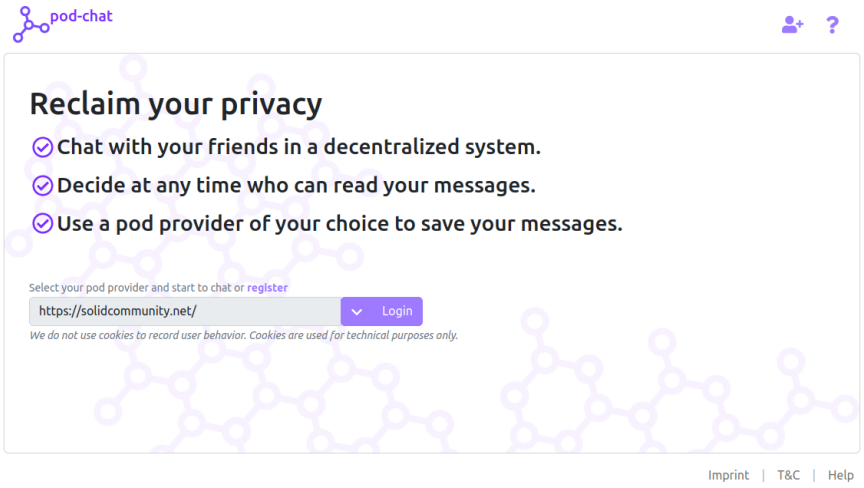

Figura 4.16: Página inicial do chat-pod.com

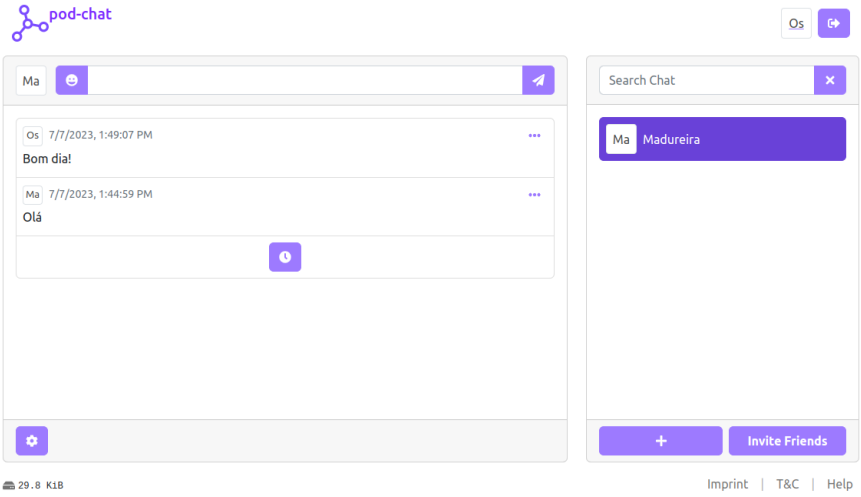

Figura 4.17: Uma face das conversas

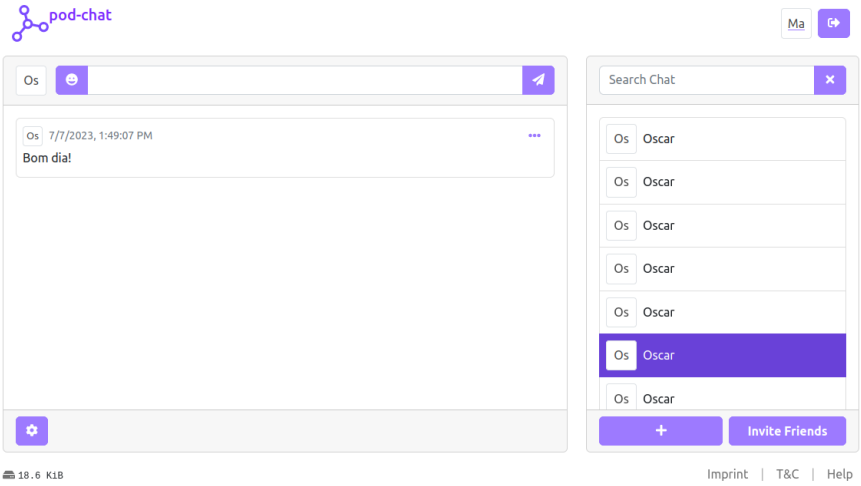

Figura 4.18: Outra face das conversas

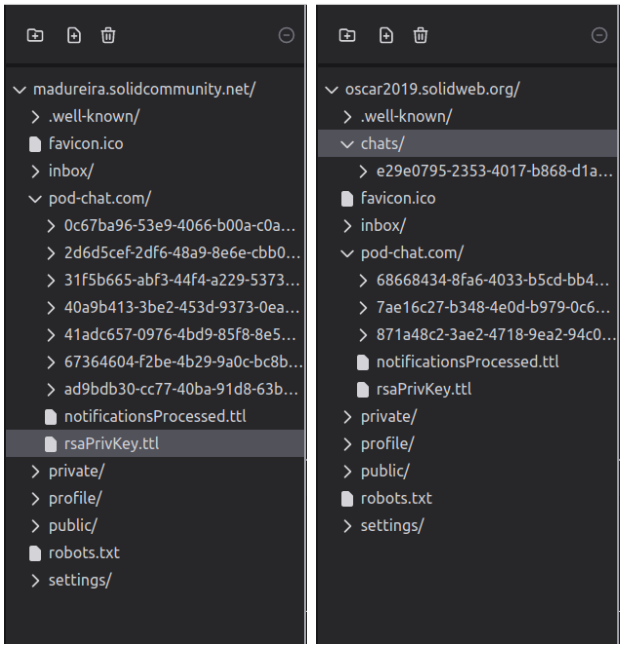

Figura 4.19: Arquivos gerados pela conversa.

4.3 Sobre implantação do CICPod Server

Devido às limitações das máquinas presentes no Departamento de Ciência da Computação, não foi possível realizar a implantação do servidor próprio do SmartUnB.ECOS. Além disso, deparamo-nos com desafios relacionados à compatibilidade do CSS (1) com as aplicações, o que nos levou a adotar o NSS como a implementação mais funcional, embora não seja o servidor padrão recomendado no site do projeto[15].

Após a conclusão da implantação do Solid Server, é fundamental realizar uma análise cuidadosa da carga de trabalho que o servidor enfrentará ao hospedar múltiplos Pods e lidar com várias requisições de dados. Além disso, será necessário adaptar os componentes para a integração com o Solid e desenvolver ou adaptar aplicações relevantes, de modo a tornar o uso dessa tecnologia cada vez mais comum e intuitivo, seguindo a tendência atual da *Web*.

Dentre as diversas possibilidades de utilização do Pod *server* no ecossistema SmartUnB.ECOS, exemplifica-se, na figura 4.20, a interação entre o Pod de um usuário (como um discente), um LMS e a plataforma de gestão de *badges*. Nesse contexto, o Pod cumpre sua finalidade ao servir como um repositório para armazenamento de dados pessoais, tanto dos *badges* quanto de outros arquivos de natureza educacional ou não. A integração do LMS (por exemplo, Moodle) com a plataforma de *badging* permite que os usuários

[15] <www.solidproject.org>

exibam seus *badges* e realizem ações com base neles dentro do ambiente de aprendizagem. Por exemplo, um aluno pode exibir seus *badges* em seu perfil do Moodle ou receber benefícios específicos com base nos *badges* conquistados, como acesso a conteúdos adicionais ou privilégios extras. A plataforma de *badging*, por sua vez, desempenha o papel de emitir, rastrear e validar os *badges*, fornecendo mecanismos para verificar a autenticidade e validade desses *badges* recebidos pelos indivíduos. A colaboração entre esses elementos promove um ambiente educacional mais dinâmico e personalizado, permitindo que os alunos tenham um maior controle sobre seus dados e desfrutem de uma experiência de aprendizado mais interativa e envolvente.

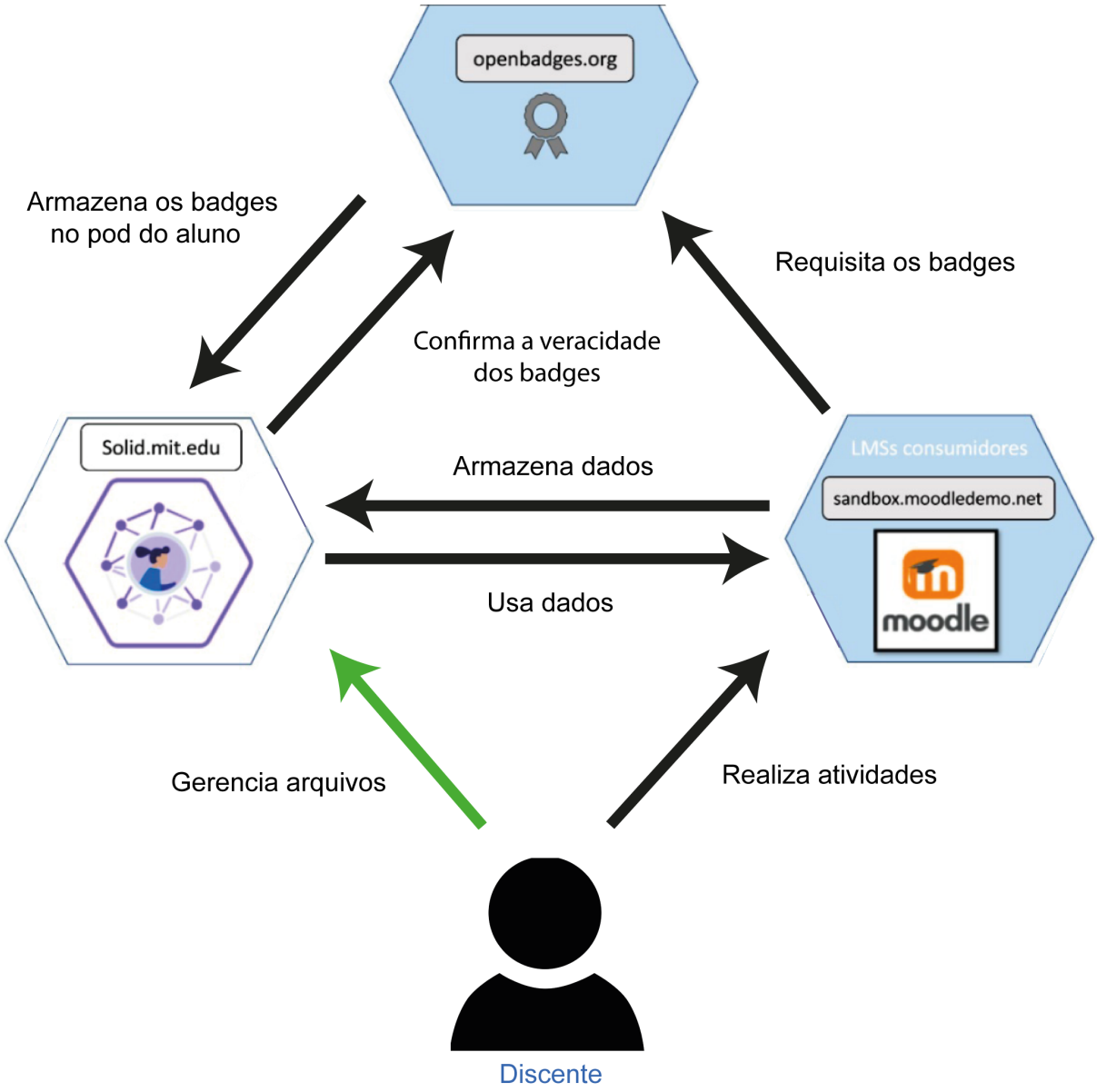

Figura 4.20: Exemplo de interação com componentes do ecossistema SmartUnB.ECOS

4.4 O Tutorial Desenvolvido

O objetivo do Tutorial, que se encontra no endereço <https://tutorial-solid.vercel.app/>, é fornecer uma introdução ao projeto Solid Pod para os usuários, oferecendo também a oportunidade de compreender melhor os processos técnicos envolvidos. Além disso, o tutorial recomenda algumas ferramentas que os leitores podem utilizar para gerenciar seus pods.

Para atingir esses objetivos, o material foi desenvolvido no formato *Web*, visando proporcionar uma experiência interativa aos usuários. O tutorial está dividido em várias seções, incluindo uma Introdução que contextualiza o projeto, bem como seções sobre a usabilidade do Pod, abrangendo os processos de Criar, Ler, Atualizar e Deletar recursos. Também são abordados tópicos como Mover o Pod e Experimentando o Pod.

É importante destacar que o material pode ser atualizado ao longo do tempo para adicionar novos conteúdos ou melhorar a forma como os dados são apresentados, garantindo que o tutorial se mantenha relevante e atualizado.

Figura 4.21: *Home Page* do tutorial feito no formato *Web*.

4.5 Introdução do Tutorial

A página de Introdução traz, logo de início, informações de como o projeto Solid foi concebido, sua autoria e as vantagens do projeto. Foi feita uma descrição do uso geral da *Web* na época da escrita e uma comparação de como a abordagem corrente afeta a experiência do usuário com os dados e como o Solid pode melhorá-la no contexto de exames hospitalares. Também é mencionado como os dados são tratados com o Solid e como se relaciona com outras tecnologias como o RDF e Linked Data.

Logo em seguida é feito uma espécie de glossário com os termos principais para o entendimento do tutorial: Solid, Pod, *WebId*, Pod *Server*, URI. Esses termos são usados em seguida para explicar outros conceitos mais complexos usados na composição composição do Solid ou relacionados a ele.

Explica-se o RDF como um conceito central para o entendimento do funcionamento do Solid, uma vez que os arquivos estão no formato *Turtle*, uma linguagem que é uma sintaxe para RDF, também explicado posteriormente. São explicados os tipos de nós possíveis no RDF, também chamados de **Termos RDF** sendo eles: IRI, Literais ou *Literals*, Nós Anônimos ou *Blank Nodes*, Triplas Citadas ou *quoted triples*; também são fornecidos alguns exemplos em imagens.

Para explicar como podemos inserir semântica em páginas *Web*, é introduzido o conceito de *Resource Description Framework in Attributes* (RDFa). Essa tecnologia permite a adição de dados estruturados de forma que as máquinas consigam ler as informações dispostas na página, já que o HTML por si só não permite que a máquina entenda o conteúdo.

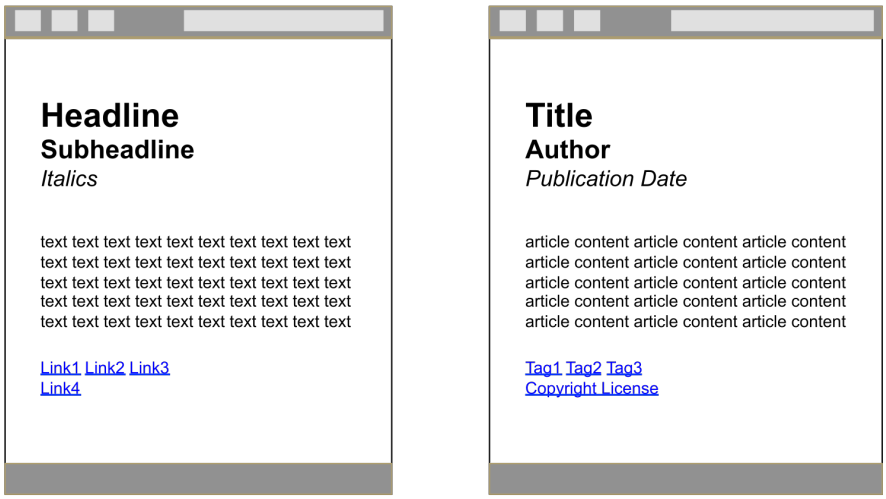

Figura 4.22: Conteúdo da página com HTML puro vs o que a máquina entende (HERMAN et al., 2015).

A explicação a respeito do Turtle está organizada de forma que o leitor possa entender os arquivos criados durante o funcionamento do Solid Pod. Assim, foca-se na sintaxe da linguagem para representar a tripla do RDF formada por sujeito, predicado e objeto. Também é explicado como os Termos RDF são representados na linguagem e como colocar comentários (PRUD'HOMMEAUX et al., 2023).

O SPARQL também é mencionado como linguagem de consulta para recuperar informações de grafos RDF. Essa linguagem permite consultas complexas e precisas em dados RDF. Por isso, é muito usada para fazer requisições em aplicações Solid-compatíveis.

Como o SPARQL possui uma sintaxe muito extensa, não foi especificado como usá-la, mas foi deixado um link para a documentação, que explica a sintaxe.

Outro termo explicado é o *Linked Data*, também mencionado nesse trabalho. A explicação menciona não só os princípios do *Linked Data*, mas também o conceito de *Linked Open Data*. Por fim, é mencionado o "esquema de classificação de 5 estrelas", que objetiva verificar o quão próximo os recursos disponíveis na *Web* estão do *Linked Data*.

Em seguida, temos o tópico do Solid na Educação. Nessa parte da introdução é dito como o Solid pode contribuir para a educação ao favorecer o *"lifelong learning"* ou aprendizado ao longo da vida por meio da concentração da bagagem educacional do estudante em um único lugar e da facilitação do compartilhamento de conteúdo educacional. É dito como o Solid pode contribuir para a pesquisa sobre questões de propriedade, privacidade e segurança de dados. É dito como o Solid pode atuar na área do aprendizado adaptativo por possibilitar que as aplicações usem dados educacionais para prover conteúdos e estratégias de ensino personalizadas. Por fim, é dito que um provedor de Solid Pods permite que os alunos conheçam e explorem um novo paradigma *Web*.

Um tópico que pode trazer o interesse tanto dos usuários em geral quanto estudantes é a questão da segurança. De forma geral, comenta-se que nem todos os provedores de Pods fornecem um verificador de força de senha e que a autenticação é feita por meio da especificação *Solid OpenID Connect* (Solid OIDC) (SOLID-OIDC,). Logo depois, é disposto um acordeão, elemento da interface do usuário que permite a expansão e a contração de conteúdo, que fornece uma explicação mais detalhada do funcionamento do processo de autenticação no ecossistema Solid. Nessa parte, a explicação se dá por meio de um exemplo fornecido na especificação que ilustra um fluxo de autenticação.

Assim como o tópico de segurança, o tópico de privacidade é algo que desperta interesse para diversos públicos, uma vez que é uma das propostas-chave do projeto. Nessa parte do texto, é indicado como é satisfeita a promessa de aprimorar e fornecer o controle da privacidade, fornecendo exemplos de tela do CSS (1) e do NSS. Assim como no tópico anterior, é exibido um acordeão que possui um exemplo com o passo a passo de um fluxo de requisição no armazenamento de outro usuário.

4.6 Usabilidade por meio do CRUD

4.6.1 Create ou Criar

O tutorial aborda o tema da usabilidade do servidor por meio da seção CRUD, iniciada pela na subseção "Criar". É mencionado que os arquivos de configuração podem ser criados

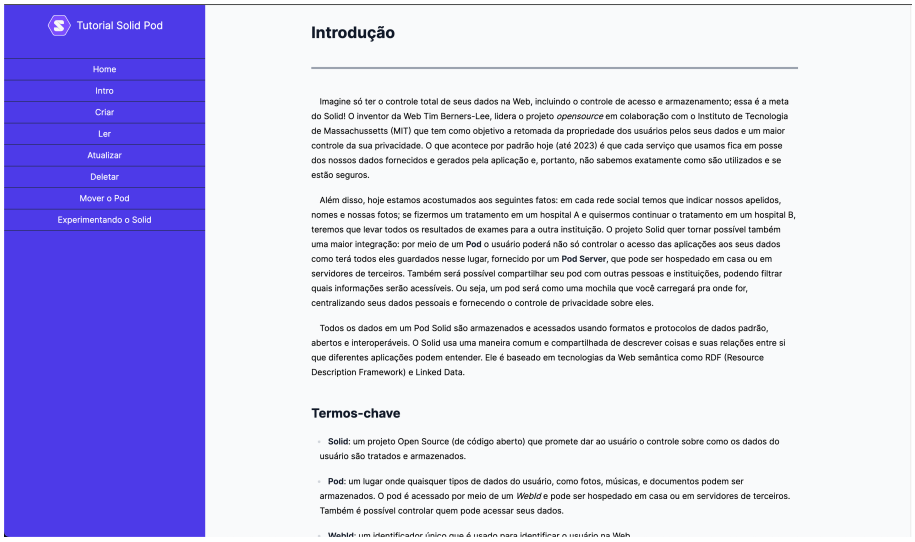

Figura 4.23: Página de Introdução

pelas aplicações ou manualmente pelo usuário. Além disso, é explicado como criar um *pod* em uma instância do CSS (1) por meio da interface.

Em seguida, são apresentadas algumas ideias sobre como gerenciar um *pod*, destacando alguns aplicativos conhecidos como *podbrowsers*. Duas opções recomendadas são o Inrupt PodBrowser[16] e o Solid Filemanager[17], mas, com o conhecimento do formato Turtle, o Penny[18] é altamente recomendado. No entanto, é importante observar que muitos dos *podbrowsers* estão em desenvolvimento, alguns foram descontinuados e outros têm pouca manutenção.

Os tópicos subsequentes se concentram em como criar recursos básicos nos *pods*, mostrando como eles são representados e apresentando as funcionalidades oferecidas pelos principais *podbrowsers*. Os processos básicos de criação incluem a criação de pastas, arquivos, contatos e *bookmarks*.

Por fim, são discutidas as requisições que podem ser feitas ao servidor para criar recursos. Esse tópico se repete com uma estrutura semelhante em todos os tópicos de usabilidade do servidor relacionados ao CRUD. São mencionados os tipos de requisições, destacando que apenas as requisições *POST* e *PUT* podem ser usadas para criar recursos. São fornecidos exemplos de requisições *curl* para os métodos *POST* e *PUT*, juntamente com algumas diretrizes (*flags*) usadas com o comando. Além disso, são apresentados exemplos reais de requisições em JavaScript para ambos os métodos de criação, com explicações e opções adequadas para cada método. Um link é fornecido para obter mais

[16] <https://podbrowser.inrupt.com/login?returnTo=%2F>
[17] <https://otto-aa.github.io/solid-filemanager/>
[18] <https://penny.vincenttunru.com/>

informações sobre os métodos no site MDN[19].

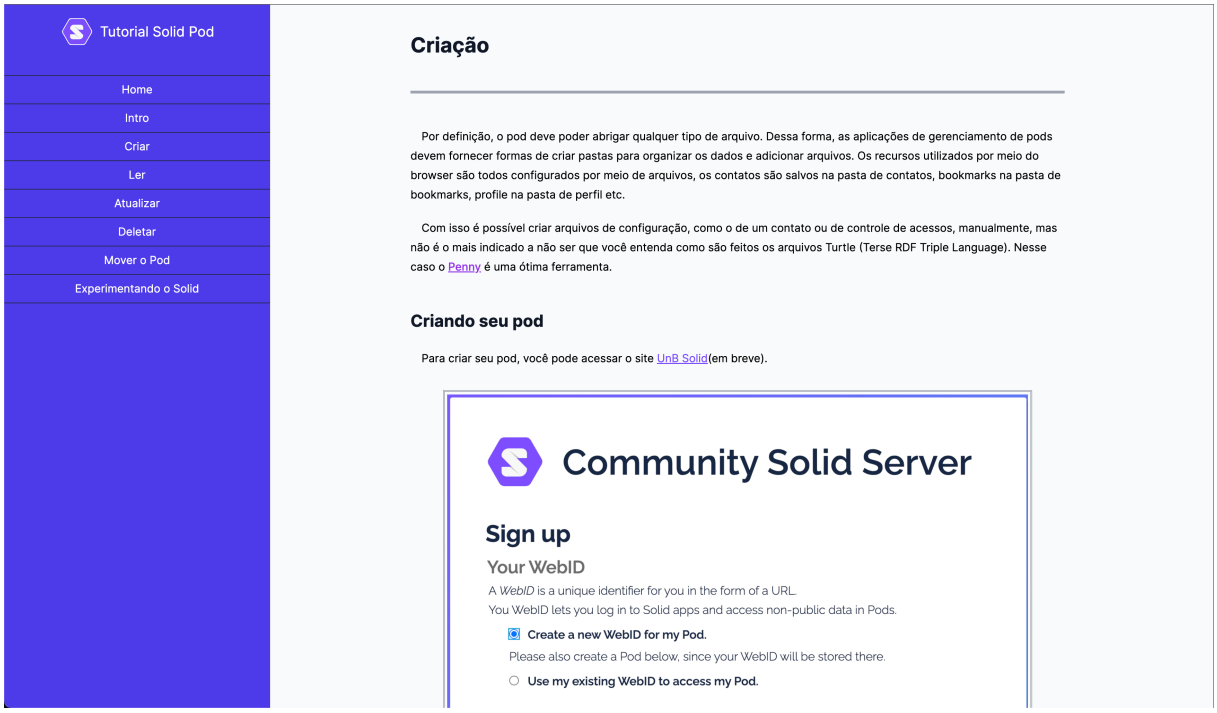

Figura 4.24: Seção Criar.

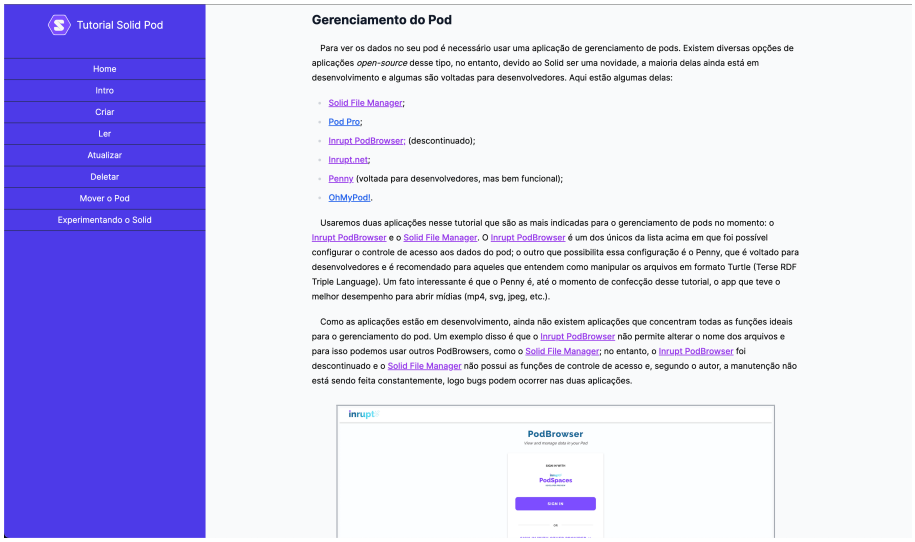

Figura 4.25: Texto sobre Gerenciamento do Pod.

[19]<https://developer.mozilla.org/pt-BR/>

4.6.2 Read ou Ler

Na seção de leitura, é abordado o suporte para diferentes tipos de arquivos nos principais *podbrowsers* recomendados. Observa-se que a maioria dessas ferramentas não consegue abrir arquivos de vídeo ou exibir arquivos *.svg*. No entanto, destaca-se que o Penny é capaz de abrir um número maior de tipos de arquivos de maneira satisfatória.

Outro aspecto discutido é o controle de acesso, que pode ser realizado tanto por meio da interface dos *podbrowsers* quanto através de arquivos *.acl* (Arquivos de Controle de Acesso, em inglês). Explica-se como o usuário pode lidar com esses arquivos para gerenciar as permissões, já que a maioria dos *podbrowsers* disponíveis não oferece funcionalidades robustas para esse fim. É fornecido um exemplo de um arquivo *.acl*, explicando as definições e permissões concedidas no documento.

Por fim, aborda-se um tema comum em todos os tópicos de usabilidade básica do servidor: as requisições. É apresentado um exemplo de uma requisição *curl* para recuperar dados de uma rota. No entanto, ressalta-se que a melhor abordagem para realizar essas requisições é por meio de código, uma vez que é necessário obter permissões adequadas na maioria dos casos. São dados exemplos de três tipos de requisições que podem ser utilizadas para recuperar dados: GET, HEAD e OPTIONS. Também é explicada a utilidade de cada um desses métodos e fornecido um link para obter mais informações sobre eles.

Essas adições ao texto oferecem mais detalhes sobre os tipos de arquivos suportados pelos *podbrowsers*, a gestão de controle de acesso por meio de arquivos *.acl* e a utilização de diferentes tipos de requisições para recuperar dados do servidor.

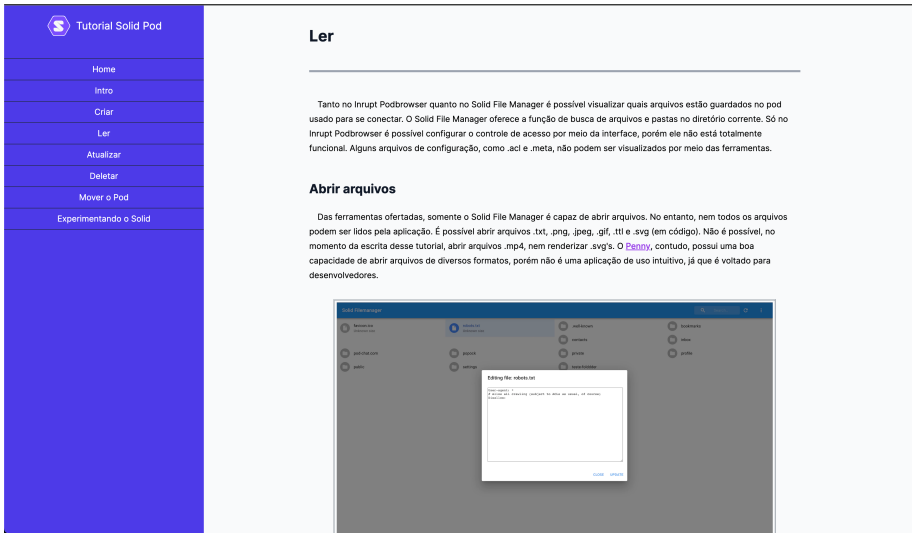

Figura 4.26: Seção Ler do Tutorial.

4.6.3 Update ou Atualizar

Nessa seção, são descritos os recursos e funcionalidades dos aplicativos de gerenciamento de Pods, destacando a capacidade de modificar e sobrescrever os arquivos armazenados.

No Solid File Manager, é destacado que os usuários têm a opção de editar determinados formatos de arquivos de texto ou código, como *.txt* e *.svg*. Além disso, é mencionado que é possível sobrescrever arquivos ao renomeá-los para um nome já existente ou adicionar arquivos com o mesmo nome de um arquivo já existente. É ressaltado que é importante ter cautela ao utilizar essa aplicação, uma vez que não há confirmação durante o processo.

Por outro lado, é mencionado que o Inrupt Podbrowser solicita confirmação antes de sobrescrever arquivos, porém não permite a edição direta de arquivos dentro da aplicação.

É mencionado que também é possível realizar alterações manuais nas informações de contato, bookmarks e controle de acesso por meio de arquivos no formato Turtle. Essa opção oferece aos usuários a capacidade de personalizar esses elementos conforme suas necessidades e preferências.

Em relação às requisições, o foco é nos métodos PUT e PATCH, que são capazes de realizar a atualização ou sobrescrita de recursos. São fornecidos exemplos de uso e é destacado que é possível criar ou alterar um recurso utilizando o método PUT, especificando o formato do conteúdo no cabeçalho da requisição.

O método PATCH é explicado como uma opção para atualizar recursos, realizando alterações nos dados já existentes, em contraste com o método PUT, que sobrescreve o recurso por completo. É fornecido um exemplo de uso do método PATCH, enfatizando a importância de utilizar o tipo de conteúdo *'text/n3'* para recursos no formato RDF.

Por fim, é mencionado que a linguagem N3 (Notation 3) é utilizada em requisições PATCH com o tipo de conteúdo *'text/n3'*. É fornecido um link para obter mais informações sobre essa linguagem.

4.6.4 Delete ou Deletar

Na seção de deleção de recursos, é mencionado que o proprietário do Pod possui permissão para deletar os arquivos em seu Pod. Também é abordado como as aplicações podem colaborar com esse processo por meio da interface e que é possível configurar permissões de deleção para terceiros. No entanto, é ressaltado que os arquivos excluídos são permanentemente removidos e não podem ser recuperados.

Em seguida, são apresentadas as requisições para deletar recursos. Além da explicação tradicional sobre as requisições e o uso do comando *curl*, essa seção destaca o método DELETE, o único método que tem a função específica de deletar. É fornecido um exemplo de requisição *curl* utilizando o método DELETE.

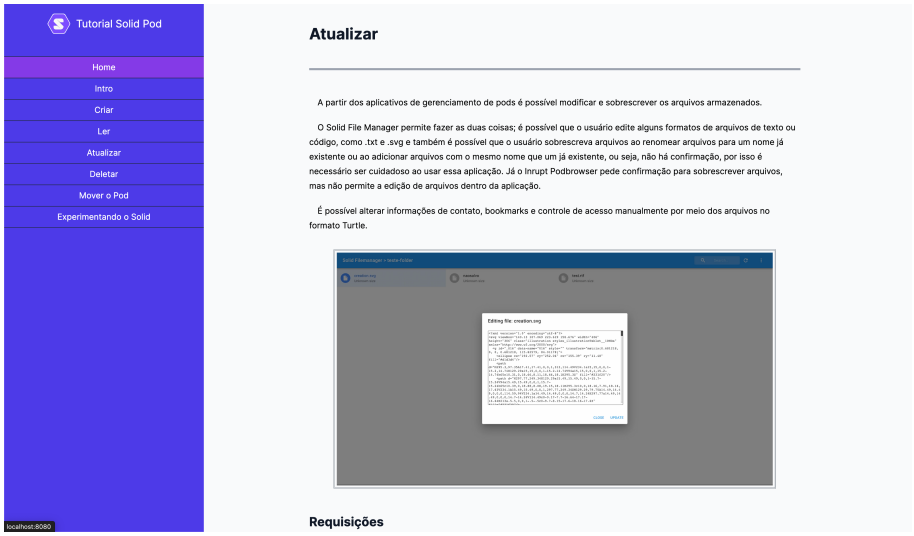

Figura 4.27: Seção Atualizar

Além disso, é dado um exemplo de como o método DELETE pode ser utilizado por meio do Javascript. Também é fornecido um link que direciona para mais informações sobre esse tipo de requisição.

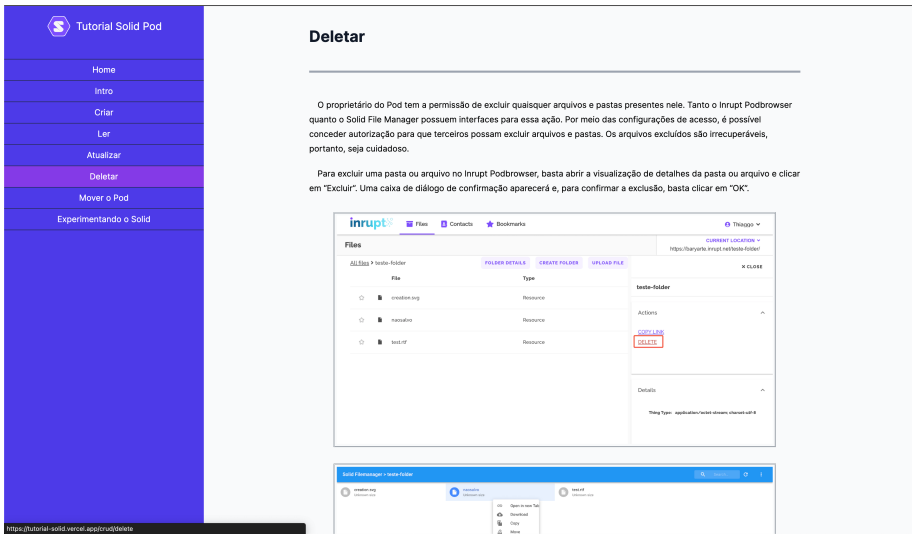

Figura 4.28: Seção Deletar

4.7 Mover o Pod

A movimentação ou exportação de um Pod para outro provedor é uma funcionalidade importante, pois afeta a capacidade dos usuários de escolherem a localidade dos seus dados e terem controle sobre eles. No entanto, ao analisar o estado atual da arte e o

projeto Solid, observou-se uma lacuna em relação a diretrizes específicas para realizar esse procedimento.

O documento que especifica o Protocolo Solid (CAPADISLI et al., 2022), que serve como referência para o projeto, não fornece instruções detalhadas sobre como mover ou exportar um Pod para outro provedor. Isso indica que cada provedor pode abordar essa funcionalidade de maneira independente, de acordo com suas próprias políticas e recursos.

Durante a pesquisa, foram encontrados provedores que oferecem a opção de exclusão do Pod, permitindo que os usuários removam completamente seus dados. No entanto, também foram encontrados provedores que não fornecem essa opção, o que pode restringir a liberdade dos usuários em relação à movimentação de seus Pods.

No caso específico da implementação escolhida, o Community Solid Server (CSS), não foi identificada uma maneira específica de exportar ou deletar o Pod. Essa limitação pode ser atribuída ao fato de que o projeto Solid ainda está em desenvolvimento ativo e algumas funcionalidades podem não estar totalmente implementadas ou disponíveis em todas as implementações do servidor.

É importante ressaltar que, até o momento da pesquisa, nenhum provedor testado ofereceu a opção de exportação dos dados do Pod. Isso destaca a necessidade de avaliar cuidadosamente as funcionalidades e políticas oferecidas por cada provedor antes de tomar decisões em relação à movimentação dos dados.

Essa falta de padronização e diretrizes específicas para a movimentação de Pods entre provedores é um aspecto importante a ser considerado na comparação entre a proposta do projeto Solid e o estado atual da arte. A comunidade Solid continua a evoluir e é possível que soluções para a movimentação de Pods sejam desenvolvidas no futuro, oferecendo aos usuários maior flexibilidade e controle sobre seus dados.

Figura 4.29: Seção Mover o Pod

4.8 Experimentando o Solid

Esta seção do trabalho descreve o teste realizado com o Solid Pod, focando no armazenamento e controle de acesso em uma aplicação de chat. As aplicações de chat selecionadas para o teste foram LiqidChat e Pod-Chat, que foram analisadas em relação à sua integração com os Solid Pods.

Os testes foram conduzidos utilizando duas implementações de provedores, o *Community Solid Server* (CSS (1)) e o *Node Solid Server* (NSS). A forma de armazenamento dos dados é determinada pela aplicação de chat utilizada, e um diagrama e um passo a passo detalhados foram fornecidos para explicar a interação entre os usuários, seus provedores e o servidor da aplicação de chat.

Durante o teste, foram identificados alguns problemas de compatibilidade entre o CSS e as aplicações de chat selecionadas. É importante ressaltar que esses problemas são compreensíveis, uma vez que o projeto Solid ainda está em desenvolvimento e a integração perfeita entre as aplicações e os provedores pode ser um desafio em estágios iniciais.

Ao comparar as aplicações de chat em relação ao armazenamento de dados e controle de acesso, constatou-se que o Pod-Chat está mais alinhado com a proposta do projeto Solid, oferecendo um controle mais igualitário aos usuários sobre seus dados. No entanto, foi observado que ambas as aplicações requerem um número significativo de permissões para funcionar, o que pode ser questionado em termos de privacidade e controle de acesso em um ambiente Solid.

Adicionalmente, foi realizada uma comparação entre o chat descentralizado e o centralizado, abordando os principais aspectos relacionados à posse dos dados e à privacidade dos usuários. Essa comparação ressalta a visão inovadora do Solid Pods, onde os usuários têm maior controle sobre seus dados e podem escolher onde e como seus dados são armazenados, em oposição ao modelo centralizado tradicional, onde os dados são de propriedade e controlados por terceiros.

Esta seção desempenha um papel fundamental ao aprofundar nossa compreensão do potencial dos Solid Pods, não apenas no contexto de aplicações de chat, mas também no entendimento da dinâmica entre as aplicações e o projeto Solid de uma forma mais abrangente. Ao destacar os desafios atuais e as oportunidades futuras para melhorias na integração e experiência do usuário, essa seção nos proporciona uma visão mais aprofundada das capacidades e possibilidades dos Solid Pods. Compreender essa dinâmica é essencial para explorar todo o potencial transformador que os Solid Pods oferecem, não apenas para aplicativos de chat, mas para uma ampla gama de soluções descentralizadas e centradas no usuário.

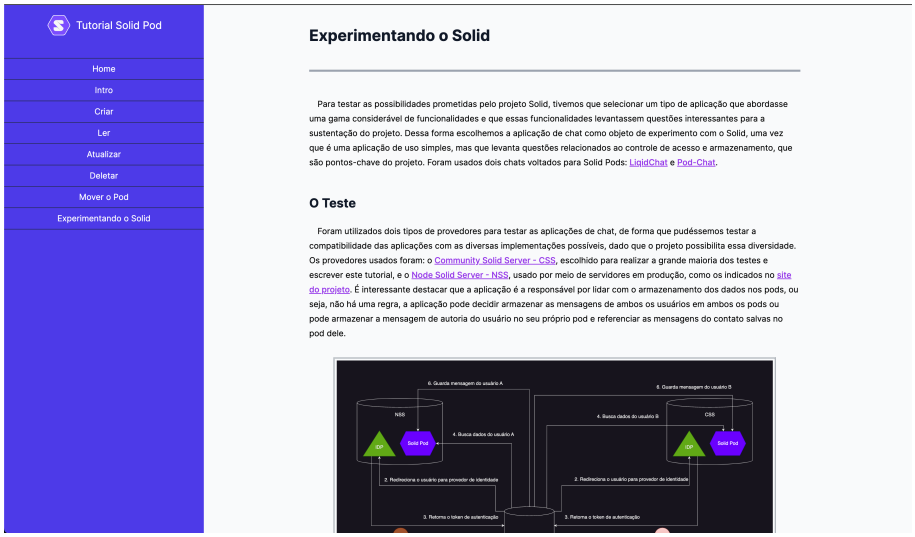

Figura 4.30: Seção Experimentando o Solid

Capítulo 5

Conclusão

5.1 Objetivos alcançados

Pode-se dividir a conclusão em duas etapas principais: Implementação do servidor; E o tutorial de desenvolvimento. Sendo este trabalho uma forma de agregar para o UnB.ECOS, é perceptível que a etapa de implementação do servidor não foi de um sucesso absoluto, uma vez que só foi alcançada a implementação *localhost*, assim não exercendo o mesmo impacto no projeto UnB.ECOS. Por outro lado, este trabalho agrega de uma forma mais densa ao propor o tutorial de utilização, ou desenvolvimento de aplicativos, do protocolo solid.

Ainda que o impacto da implementação *localhost* pudesse ter sido maior a partir dos teste realizados foi percebido a não completude do protocolo *solid*. Isso pela falta de congruência entre o versionamento de das aplicações servidores e aplicações clientes, fato que é perceptível pela falta de compatibilidade entre as aplicações e por alguns códigos de aplicações, explicitamente, aceitares somente um servidor, o que foge da proposta do protocolo *solid*. Além deste existem inúmeros pontos de ressaltam a imaturidade do protocolo, a falta o cumprimento de promessas e a falta de abordagem de outras questões, ficando em evidência a incapacidade de mover o *Pod* (uma vez que os dados estão sobre o controle do usuário, o usuário teve ter total controle sobre o que ele deve fazer sobre esses dados), e a necessidade de muitas aplicações de terem o controle total do *Pod* (isso pode gerar um desconforto de dados para os usuários, principalmente para os usuários inexperientes, que podem permitir que uma aplicações maliciosa roube todos os seus dados).

E mesmo que haja incongruências, descoberta através de teste, não é justificável anular toda a proposta do protocolo. Então faz-se necessário um tutorial capaz de guiar os futuros alunos para aplicar a proposta do solid Pod, gerando mais aplicações e promovendo novas

ideias capazes de transformar o modo como o atual mercado de armazenamento de dados é pensado. pados centralizados.

Apesar de ser uma boa proposta, a capacidade de implementação deve ser questionada. Pois não há muito conteúdo que fala sobre o protocolo *solid* — reduzindo os locais de aprendizagem do protocolo em comparação com a quantidade de locais que permitem o aprendizado da forma centralizada de desenvolvimento —, a velocidade de crescimento do protocolo é ínfima se comparado ao crescimento de aplicações da *Web* descentralizada — o que dificulta ainda mais a transição do protocolo, já que a dificuldade de transcrever um sistema é muito grande, tendo como exemplos os banco, que até hoje em dia utilizam COBOL como linguagem de programação —, e por fim, o empecilho das *big techs* — elas dificultariam a transição para um novo protocolo, pois hoje elas detém muitos dados do usuário.

Assim, pode-se concluir que a melhor forma de utilizar o pod nesse atual momento é agregando a lista confiável de aplicações, lista que pode ser disponibilizada pela própria página do projeto *solid*:

- Se for necessário identificar os usuários, eles devem ser capazes de fazer login usando seu WebID e apontando para o Provedor de Identidade de sua escolha;
- Os dados consumidos pelo aplicativo devem ser obtidos dos Solid Pods, se possível;
- Os dados gerados pelo aplicativo devem ser armazenados nos Solid Pods;
- A interação entre o aplicativo, os pods e o(s) Provedor(es) de Identidade deve estar em conformidade com a especificação Solid.

Ou por instituições de confiança do usuário, como seu hospital, sua escola ou a empresa em que ele trabalha, vale ressaltar que para dados que o usuário não pode alterar diretamente deve-se usar algum sistema de assinatura.

5.2 Trabalhos Futuros

Mesmo com a finalização deste trabalho, ainda há uma grande quantidade de estudos a serem feitos, pois este conteúdo é jovem e promissor.

O principal trabalho que pode ser feito a partir deste é a hospedagem do servidor, bem como a sua devida integração. Visto que a hospedagem do servidor pode agrega em grande proporção em toda a ideia do projeto SmartUnB.ECOS, além de se tornar de fato uma caixa de areia que permite que os alunos possam acessar, descobrir e assimilar as ideias de descentralização.

Outros trabalhos que podem ser feitos são o estudo do protocolo e de suas implementações (afim de sugerir melhorias para o protocolo, e fazer com que as implementações disponíveis se aproximem cada vez mais do mesmo), e a criação de novos aplicativos, como em (DUDA, 2023), assim como o estudo de ética em aplicações integradas ao Solid, semelhante à análise de (CRUVINEL; MORAIS, 2023), que aborda LMSs. E, por fim, a manutenção do tutorial, pois devido à imaturidade do protocolo, podem ocorrer alterações e o tutorial pode se tornar defasado.

Referências Bibliográficas

ALABDULWAHHAB, F. A. Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation. In: *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*. [S.l.: s.n.], 2018. p. 1–4. 1

BAHRI, L.; CARMINATI, B.; FERRARI, E. Decentralized privacy preserving services for online social networks. *Online Social Networks and Media*, v. 6, p. 18–25, 2018. ISSN 2468-6964. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2468696417301040>. 11

BERNERS-LEE, T. Information Management: A Proposal. 1990. Disponível em: <https://chnm.gmu.edu/digitalhistory/links/pdf/finalthoughts/f_1.pdf>. 16

BERNERS-LEE, T. *Linked Data*: Design issues. 2006. W3C Website. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>. 10

BERNERS-LEE, T. *The next web*: Tim berners-lee | ted2009. 2009. TED Talk. Disponível em: <https://www.ted.com/talks/tim_berners_lee_the_next_web>. 18

BERNERS-LEE, T. *DWeb Summit 2016 Keynote: Tim Berners-Lee*. 2016. Internet Archive. Disponível em: <http://archive.org/details/DWebSummit2016_Keynote_Tim_Berners_Lee>. 1

BERNERS-LEE, T.; MASINTER, L. M.; FIELDING, R. T. *Uniform Resource Identifiers (URI): Generic Syntax*. [S.l.], 1998. Num Pages: 40. Disponível em: <https://datatracker.ietf.org/doc/rfc2396>. 8

BERNERS-LEE, T.; MASINTER, L. M.; MCCAHILL, M. P. *Uniform Resource Locators (URL)*. [S.l.], 1994. Num Pages: 25. Disponível em: <https://datatracker.ietf.org/doc/rfc1738>. 8

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data: The story so far. *International Journal on Semantic Web and Information Systems*, v. 5, p. 1–22, 07 2009. 10

BOERBOOM, C. Cambridge Analytica: The Scandal on Data Privacy. *Augustana Center for the Study of Ethics Essay Contest*, jan. 2020. Disponível em: <https://digitalcommons.augustana.edu/ethicscontest/18>. 20

BRASIL, W. *Web Semântica*: Padrões web. 2011. W3C Brasil Website. Disponível em: <https://www.w3c.br/Padroes/WebSemantica>. 9

CAPADISLI, S. et al. *Solid Protocol*. 2022. Version 0.10.0. Disponível em: <https://solidproject.org/TR/protocol>. 57

CERN. *A short history of the Web*: The birth of the web. 2023. CERN Website. Disponível em: <https://home.web.cern.ch/science/computing/birth-web/short-history-web>. 18

CRUVINEL, A. V. C. V.; MORAIS, L. G. Aplicações Educacionais na Web Descentralizada: Uma Investigação em Perspectiva da Ética Computacional e da Privacidade de Dados à Luz do LMS Moodle. Universidade de Brasília, Brasília, 2023. (em andamento). 62

DRURY, B. et al. A survey of semantic web technology for agriculture. *Information Processing in Agriculture*, v. 6, n. 4, p. 487–501, dez. 2019. ISSN 2214-3173. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214317318302580>. 9

DUDA, J. M. S. Uma investigação sobre o projeto solid: da prospecção para ecossistema educacional ao desenvolvimento de aplicações como prova de conceito. Universidade de Brasília, Brasília, 2023. 57 f., il. 62

DüRST, M.; SUIGNARD, M. *Internationalized Resource Identifiers (IRIs)*. 2005. RFC Editor. Disponível em: <https://www.rfc-editor.org/rfc/rfc3987>. 8, 9

FEIGENBAUM, L. et al. *SPARQL 1.1 Protocol*: W3c recommendation 21 march 2013. 2013. W3C Recommendation. Disponível em: <https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/>. 22

FIELDING, R. T. *Relative Uniform Resource Locators*. [S.l.], 1995. Num Pages: 16. Disponível em: <https://datatracker.ietf.org/doc/rfc1808>. 8

GHAYVAT, H. et al. SHARIF: Solid Pod-Based Secured Healthcare Information Storage and Exchange Solution in Internet of Things. *IEEE Transactions on Industrial Informatics*, v. 18, n. 8, p. 5609–5618, ago. 2022. ISSN 1941-0050. Conference Name: IEEE Transactions on Industrial Informatics. 14

GROUP, T. W. S. W. *SPARQL 1.1 Overview*: Terse rdf triple language. 2013. W3C Recommendation. Disponível em: <https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/>. 22

HARRIS, S.; SEABORNE, A.; GROUP, T. W. S. W. *SPARQL 1.1 Federated Query*: W3c recommendation 21 march 2013. 2013. W3C Recommendation. Disponível em: <https://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/>. 22

HERMAN, I. et al. *RDFa 1.1 Primer - Third Edition*: Rich structured data markup for web documents. 2015. W3C Note. Disponível em: <https://www.w3.org/TR/rdfa-primer/>. x, 22, 50

HIREMATH, B. K.; KENCHAKKANAVAR, A. Y. An Alteration of the Web 1.0, Web 2.0 and Web 3.0: A Comparative Study. v. 2, n. 4, 2016. 4

HISTORY. 2023. Disponível em: <https://www.w3.org/about/history/>. 18

JANEV, V.; VRANEš, S. Applicability assessment of Semantic Web technologies. *Information Processing & Management*, v. 47, n. 4, p. 507–517, jul. 2011. ISSN 03064573. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0306457310000920>. 9

JANSSEN, H. et al. Decentralized data processing: personal data stores and the GDPR. *International Data Privacy Law*, v. 10, n. 4, p. 356–384, jan. 2021. ISSN 2044-3994, 2044-4001. Disponível em: <https://academic.oup.com/idpl/article/10/4/356/6054280>. 19

JUVITO, L. de A.; SOARES, R. T. Um projeto de Metaversidade: ampliando possibilidades para cursos de Computação. Universidade de Brasília, Brasília, 2023. (em andamento). 33

LAAL, M.; SALAMATI, P. Lifelong learning; why do we need it? *Procedia - Social and Behavioral Sciences*, v. 31, p. 399–403, 2012. ISSN 18770428. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1877042811030023>. 6

MANSOUR, E. et al. A Demonstration of the Solid Platform for Social Web Applications. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016. (WWW '16 Companion), p. 223–226. ISBN 978-1-4503-4144-8. Disponível em: <https://doi.org/10.1145/2872518.2890529>. ix, 14, 15, 19

MOATS, R. *URN Syntax*. [S.l.], 1997. Num Pages: 8. Disponível em: <https://datatracker.ietf.org/doc/rfc2141>. 8

MOREIRA, F. M. et al. Tecnologias da Web Semântica para a recuperação de dados agrícolas: um estudo sobre o International Information System of the Agricultural Science and Technology (AGRIS). *Em Questão*, p. 173–192, maio 2015. ISSN 1808-5245. Disponível em: <https://seer.ufrgs.br/index.php/EmQuestao/article/view/50317>. 9

NUPUR, C. *World wide web and its journey from web 1.0 to web 4 - IJCSIT*. International Journal of Computer Science and Information Technologies, 2014. Disponível em: <https://docslib.org/doc/2952074/world-wide-web-and-its-journey-from-web-1-0-to-web-4-0>. 4

NóBREGA, G. M. d.; SILVA, G. T. d.; SILVA, T. V. R. Um projeto estruturante para orientações de TCC em cursos de computação: que oportunidades para IHC? In: *Anais do Workshop sobre Educação em IHC*. SBC, 2022. p. 19–24. ISSN: 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/weihc/article/view/22854>. ix, 3, 20

PROJECT, T. S. *Solid Protocol*: Web access control. 2021. Solid Project Technical Report. Disponível em: <https://solidproject.org/TR/protocol#web-access-control>. 19

PROJECT, T. S. *About Solid · Solid*. [s.d.]. Disponível em: <https://solidproject.org/about>. 2

PRUD'HOMMEAUX, E. et al. *RDF 1.1 Turtle*: Terse rdf triple language. 2014. W3C Recommendation. Disponível em: <https://www.w3.org/TR/turtle/#turtle-literals>. 21, 24, 25

PRUD'HOMMEAUX, E. et al. *RDF 1.2 Turtle*: Terse rdf triple language. 2023. W3C Working Draft. Disponível em: <https://www.w3.org/TR/rdf12-turtle/>. 50

RAJAGOPAL, K. Personal Learning Environments as socio-technical systems: does decentralised data finally give us the right balance? *Revista de Educación a Distancia (RED)*, v. 23, n. 71, jan. 2023. ISSN 1578-7680. Number: 71. Disponível em: <https://revistas.um.es/red/article/view/526851>. ix, 15, 16

RENJIFO, D. *Inventor of the world wide web wants us to reclaim our data from tech giants | CNN Business*. 2022. Disponível em: <https://www.cnn.com/2022/12/16/tech/tim-berners-lee-inrupt-spc-intl/index.html>. 18

SAMBRA, A. V. et al. Solid : A platform for decentralized social applications based on linked data. In: . [s.n.], 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:49564404>. 18, 20

SHADBOLT, N.; BERNERS-LEE, T.; HALL, W. The Semantic Web Revisited. *IEEE Intelligent Systems*, v. 21, n. 3, p. 96–101, jan. 2006. ISSN 1941-1294. Conference Name: IEEE Intelligent Systems. 18

SINGH, M. *Web creator Tim Berners-Lee's startup Inrupt raises $30 million*. 2021. Disponível em: <https://techcrunch.com/2021/12/09/tim-berners-lee-inrupt-fundraise/>. 18

SOLID-OIDC. Disponível em: <https://solidproject.org/TR/oidc#biblio-solidoidcprimer>. 51

STORY, H. et al. *WebID 1.0: Web Identity and Discovery*: Incubator group report. 2011. W3C Incubator Group Report. Disponível em: <https://www.w3.org/2005/Incubator/webid/spec/identity/>. 19, 20

SUTIKNO, T.; AISYAHRANI, A. I. B. Non-fungible tokens, decentralized autonomous organizations, Web 3.0, and the metaverse in education: From university to metaversity. *Journal of Education and Learning (EduLearn)*, v. 17, n. 1, p. 1–15, fev. 2023. ISSN 2302-9277. Number: 1. Disponível em: <http://edulearn.intelektual.org/index.php/EduLearn/article/view/20657>. 13

THILAKARATHNA, K. et al. Mobitribe: Cost efficient distributed user generated content sharing on smartphones. *IEEE Transactions on Mobile Computing*, v. 13, n. 9, p. 2058–2070, Sep 2014. 11

YEUNG, C.-m. et al. Decentralization: The Future of Online Social Networking. v. 2, maio 2011. 1

Apêndice A

json de configuração de pod

```
{
  "@context": "https://linkedsoftwaredependencies.org/bundles
     /npm/@solid/community-server/^5.0.0/components/context.
     jsonld",
  "import": [
    "css:config/app/init/default.json",
    "css:config/app/main/default.json",
    "css:config/app/setup/disabled.json",
    "css:config/app/variables/default.json",
    "css:config/http/handler/default.json",
    "css:config/http/middleware/websockets.json",
    "css:config/http/server-factory/https-websockets.json",
    "css:config/http/static/default.json",
    "css:config/identity/access/public.json",
    "css:config/identity/email/default.json",
    "css:config/identity/handler/default.json",
    "css:config/identity/ownership/token.json",
    "css:config/identity/pod/static.json",
    "css:config/identity/registration/enabled.json",
    "css:config/ldp/authentication/dpop-bearer.json",
    "css:config/ldp/authorization/allow-all.json",
    "css:config/ldp/handler/default.json",
    "css:config/ldp/metadata-parser/default.json",
    "css:config/ldp/metadata-writer/default.json",
    "css:config/ldp/modes/default.json",
    "css:config/storage/backend/file.json",
```

```
25      "css:config/storage/key-value/resource-store.json",
26      "css:config/storage/middleware/default.json",
27      "css:config/util/auxiliary/no-acl.json",
28      "css:config/util/identifiers/suffix.json",
29      "css:config/util/index/default.json",
30      "css:config/util/logging/winston.json",
31      "css:config/util/representation-conversion/default.json",
32      "css:config/util/resource-locker/file.json",
33      "css:config/util/variables/default.json"
34    ],
35    "@graph": []
36  }
```