



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

**Uma Investigação Sobre o Projeto Solid: da
Prospecção para Ecosystema Educacional ao
Desenvolvimento de Aplicações como Prova de
Conceito**

João Marcos Schmaltz Duda

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

Orientadora
Prof.a Dr.a Germana Menezes da Nóbrega

Brasília
2023

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

D844i Duda, João Marcos Schmaltz
Uma Investigação Sobre o Projeto Solid: da Prospecção para
Ecossistema Educacional ao Desenvolvimento de Aplicações
como Prova de Conceito / João Marcos Schmaltz Duda;
orientador Germana Menezes da Nóbrega. -- Brasília, 2023.
46 p.

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

1. Solid. 2. privacidade de dados. 3. desenvolvimento.
4. smartUnB.ECOS. 5. PODs. I. da Nóbrega, Germana Menezes,
orient. II. Título.

Agradecimentos

Aos meus pais, amigos e família, obrigado pelo apoio contínuo durante o curso. À minha orientadora, obrigado por me auxiliar nesta etapa final do curso, com a escolha do tema e orientação durante o desenvolvimento. Aos meus professores, obrigado por todo conhecimento. Ao professor Wilson Henrique Veneziano, obrigado por seu encorajamento para seguir na área de desenvolvimento durante sua matéria de Desenvolvimento de Aplicativos.

Obrigado. Sem vocês não teria chegado até aqui.

Resumo

A aquisição de dados na Web por parte de empresas é enorme, inclusive em ambientes educacionais, onde algumas plataformas de cursos online coletam uma variedade de dados de cada aluno ao longo da duração do curso. Essa grande quantidade de dados coletados pode ser armazenada e utilizada sem que os usuários sequer saibam quais dados seus estão sendo utilizados. Neste sentido, o projeto Solid existe para trazer o controle dos dados de volta aos usuários, e o ecossistema *smartUnB.ECOS*, um conjunto de sistemas sociais-educacionais voltado para a comunidade do Departamento de Ciência da Computação (CIC), pretende fazer uso desta tecnologia. Este trabalho visa cumprir dois objetivos: primeiro, avaliar a adoção da tecnologia Solid pela sociedade. Segundo, investigar como aplicações Solid poderiam agregar um contexto educacional como o do *smartUnB.ECOS*, avaliando a experiência de desenvolvimento com esta tecnologia. Para tanto, mapeamos algumas aplicações Solid-compatíveis, e também desenvolvemos duas aplicações próprias para avaliar como seria o processo de implementar Solid ao *smartUnB.ECOS*. Observamos que a adoção da tecnologia vem aumentando ao longo dos últimos 5 anos, e também algumas limitações de desenvolvimento das aplicações. Concluimos que a tecnologia pode ser mais divulgada para que seja mais adotada, mas que hoje já é possível desenvolver com Solid.

Palavras-chave: PODs, Solid, privacidade de dados, smartUnB.ECOS, desenvolvimento

Abstract

Data collection on the Web by tech companies is huge, even in educational environments, with some online courses platforms collecting a wide variety of data from each student over the duration of said course. Such data amounts can be stored and used without users even knowing that their data is being used. On this topic, Solid project comes to bring control of data back to the users, and the *smartUnB.ECOS* ecosystem, a set of social-educational systems geared towards the Department of Computer Science (CIC)'s community, aims to make use of this technology. This paper aims to fulfill two goals: first is to evaluate the overall adoption of the Solid technology, and second is to ponder on how Solid applications could add to an educational context such as that of *smartUnB.ECOS*, evaluating the development experience with this technology. As such, we've mapped some Solid-compatible applications, and we've also developed two custom applications to evaluate how would the process of implementing Solid to *smartUnB.ECOS*. We've noticed that the technology adoption has been on the rise over the past 5 years, and also some development limitations of the applications. We conclude the technology could be announced more in order to be more adopted, but it is already possible to develop with Solid.

Keywords: PODs, Solid, data privacy, smartUnB.ECOS, development

Sumário

1	Introdução	1
1.1	Contextualização	1
1.2	Motivação	2
1.3	Objetivos e Questão de Pesquisa	3
1.4	Organização deste documento	3
2	Fundamentação	5
2.1	Do lado do problema: Privacidade de dados educacionais	5
2.2	Do lado da solução proposta: Solid, por uma Web descentralizada	6
2.3	smartUnB.ECOS: aprendendo sobre o ecossistema de aprendizagem e ensino	8
2.4	Tecnologias Utilizadas	10
3	Aplicações em Solid	13
3.1	Provedor POD	13
3.2	Aplicações	14
3.2.1	Curso de Arquitetura de Software da Universidade de Oviedo	14
3.2.2	Media Kraken	15
3.2.3	Notepod	16
3.2.4	PodPro	16
3.2.5	Sigmafied	16
3.2.6	graphMetrix	16
3.2.7	Pojeto do sistema de saúde britânico - NHS	17
3.2.8	POD Servers do governo de Flandres	17
3.2.9	My PDS - PODs da BBC com diferentes perfis	17
3.2.10	karamel	17
3.2.11	UZ BRUSSEL RE-USE MEDICAL	17
3.2.12	Linckr	18
3.2.13	Konsolidate	18
3.2.14	VITO (BIBOPP)	18

3.2.15 We Are	18
4 Desenvolvimento dos aplicativos	20
4.1 Descrição das aplicações	20
4.2 Implementação Solid	25
4.3 Testes alfa com as aplicações desenvolvidas	30
4.3.1 RUview	30
4.3.2 Tutor	31
5 Resultados	36
5.1 Adoção	36
5.2 Desenvolvimento	39
6 Conclusão	42
Referências	43
Apêndice	45
A Links das aplicações desenvolvidas	46

Lista de Figuras

2.1	Diagrama de Descentralização do Solid ¹	7
2.2	Arquitetura 2021 smartUnB.ECOS [1].	9
4.1	Diagrama Funcionamento RUview.	21
4.2	Diagrama Funcionamento Tutor.	23
4.3	Diagrama da Estrutura das principais funções do Tutor.	23
4.4	Diagrama Exemplificação de Interoperabilidade.	24
4.5	Tela de administrador do RUview.	31
4.6	Tela com os pratos do dia do RUview.	32
4.7	Tela de edição da agenda do Tutor.	33
4.8	Tela solicitação de reunião na agenda do amigo do Tutor.	34
4.9	Tela com as reuniões agendadas do Tutor.	35
5.1	Gráfico Uso Comercial.	37
5.2	Gráfico Distribuição por País.	37
5.3	Gráfico Tipos de Serviços.	38
5.4	Gráfico Projetos por Anos.	38

Lista de Tabelas

3.1	Aplicações Solid.	19
4.1	STATUS dias da agenda.	22
4.2	STATUS de compromissos.	23

Lista de Abreviaturas e Siglas

AGR Agent/Group/Role.

CIC Departamento de Ciência da Computação.

FERPA Family Educational Rights and Privacy Act.

GDPR General Data Protection Regulation.

JSON JavaScript Object Notation.

LGPD Lei Geral de Proteção de Dados.

LMS Learning Management Systems.

LRS Learning Records Store.

LTI Learning Tools Interoperability.

MOOCs Massively Open Online Courses.

NPM Node Package Manager.

POD Personal Online Datastore.

RDF Resource Description Framework.

REST REpresentational State Transfer.

RU Restaurante Universitário.

SGBD Sistema Gerenciador de Banco de Dados.

UI User Interface.

UnB Universidade de Brasília.

URI Uniform Resource Identifier.

Capítulo 1

Introdução

1.1 Contextualização

Em um dia comum, uma pessoa acessando a Internet gera, em média, 1.7 MB de dados por segundo, ou 146,880 MB por dia por pessoa [2]. Esses dados são tudo que esta pessoa faz online: qual computador usa, qual *site* acessa, que horas liga e desliga sua máquina, quantas propagandas ela vê antes de desistir de algum *website*, além de dados que podem ser utilizados para identificá-la, como *e-mails*, geolocalização, fuso horário, idioma preferencial, entre outros [3]. Essa quantidade massiva de dados, ou *big data*, como é conhecida, pode ser utilizada para análise de padrões e customização da experiência de usuários na Web, provendo conveniência mas, também, levantando preocupações de ética e privacidade [4]. Todos os dias entregamos dados que podem ser úteis para grandes empresas, algoritmos de aprendizagem de máquina [5], campanhas publicitárias ou até mesmo para pessoas mal intencionadas [6]. Dados coletados e eventualmente transformados em informação vêm sendo considerados os recursos mais valiosos no mundo [7].

No ambiente educacional não é diferente. Alunos fornecem seus dados pessoais para escolas e universidades a todo momento. Plataformas de ensino e de cursos online estão sempre coletando dados para melhorar seus serviços. Existem plataformas de ensino que coletam até 20GB de dados de seus usuários ao longo de um curso [8]. Mesmo levando em conta que existem leis que restringem a coleta e o uso destes dados sem o consentimento do usuário, e que essas coletas possam desejar apenas a melhoria do serviço provido, ainda devem ser consideradas a grande quantidade de aplicações de ensino que um membro da comunidade acadêmica utiliza ao longo de sua formação, como *Microsoft Teams*, *Google Classroom*, *Moodle*, *Coursera*, *edX* [8]. Com cada uma dessas aplicações coletando dados (muitas vezes repetidos) sem o total conhecimento por parte do usuário sobre como serão utilizados, é fácil perceber como o mesmo não tem controle sobre os seus próprios dados.

Neste sentido, a idealização do *smartUnB.ECOS* [1] pela doutora Germana Menezes da Nóbrega traz ao debate um ecossistema de ensino formado por diferentes aplicações, em sua maior parte independentes. Trata-se de um conjunto de sistemas e ferramentas de e-learning, visando promover um retorno de projetos e sistemas úteis para o uso e aprendizado dos estudantes do Departamento de Ciência da Computação (CIC). Com essa quantidade de diferentes aplicações sob um mesmo ambiente, é de se imaginar que tenham muitos dados gerados por cada aluno em cada aplicação diferente, fora os dados redundantes que cada aplicação deve armazenar sobre cada estudante. Podemos supor que os problemas discutidos acima também se aplicam a este ecossistema; e essa suposição estaria correta senão pela última parte desse ecossistema: um *POD-Server* abrigando PODs para lidar com os dados de todas as aplicações, PODs esses pessoais e sob total controle de cada usuário. Desta forma, os estudantes sabem exatamente quais dados são utilizados por quais aplicações e com qual finalidade, podendo ser revogados ou atualizados a qualquer momento, e com suas mudanças ecoando em todas as aplicações do ecossistema, graças ao projeto Solid elaborado pelo criador da Web, Sir Tim Berners-Lee [9].

1.2 Motivação

A pandemia de COVID-19 mudou o planeta de várias formas, afetando a saúde física e mental de milhões de pessoas [10], a economia [11], a política [12, 13] e até a cultura [14]. Até a data de escrita desta monografia, ainda estamos recuperando dos efeitos dela. A Educação, um dos pilares de qualquer sociedade bem sucedida, foi uma das áreas mais afetadas pela pandemia, impedindo o contato entre alunos e professores de todas as bases educacionais, resultando em um escopo que ainda não está totalmente claro do quanto prejudicada esta área realmente foi quando se leva em conta a educação interrompida de crianças e formações atrasadas e potencialmente menos capacitadas para o mercado de trabalho do lado de universidades [15].

Para tentar controlar o dano que a pandemia causava na educação, diversos países remeteram à adoção da educação remota, ou educação a distância. Isso resultou em várias plataformas de apoio ao *e-learning* sendo inauguradas e expandidas, como Microsoft Teams, Zoom, Google Classrooms entre outras [16]. Cada uma dessas plataformas e ferramentas têm suas próprias formas de coletar e processar dados (muitas vezes os mesmos dados/dados redundantes) de seus usuários, e mesmo com avanços em leis de privacidade e proteção de dados como a Lei Geral de Proteção de Dados (LGPD), esses alunos e professores usuários dessas plataformas que se vêem compelidos a utilizá-las dado o contexto em que vivemos não possuem total controle sobre seus dados.

É clara a mudança deixada pela pandemia na educação. Embora não substituam o aprendizado presencial, ela propulsionou a adoção de tecnologias e sistemas de e-learning que são ferramentas que agregam na educação. Ainda que fosse possível voltar a como a sociedade era antes da pandemia, a adoção e utilização dessas tecnologias não irão embora, então nos resta adereçar as falhas desses sistemas e incorporá-los como ferramentas padrões no ensino.

1.3 Objetivos e Questão de Pesquisa

Este estudo tem dois objetivos maiores. O primeiro é pesquisar, listar e estudar aplicações publicadas que, de alguma forma, fazem uso do Solid em suas implementações e, com isso, reunir em um único lugar uma coleção de aplicações que servirão tanto de referência e inspiração para outras aplicações Solid como de um guia para que saibamos o quão evoluídos estão os estudos entorno do Solid atualmente. A questão é avaliar a adoção da tecnologia Solid.

O segundo objetivo é o de desenvolver dois aplicativos com Solid de modo a demonstrar a interoperabilidade da Web descentralizada, característica esta que será de grande proveito para o *smartUnB.ECOS*. Estes dois aplicativos devem ser distintos e não possuir nenhuma forma de comunicação entre eles para que a demonstração seja completa e legítima, provando sua utilidade em contexto social; e com utilidade para alunos (em especial alunos da UnB), para que os resultados sejam aplicáveis em um contexto educacional. Sendo assim, os aplicativos propostos são uma aplicação de avaliação da comida do Restaurante Universitário (RU) daquele dia, chamado RUview, e uma aplicação de agendamento de reuniões para estudos, chamado Tutor. A questão é avaliar como aplicações Solid poderiam agregar a um contexto educacional e a experiência de desenvolvimento com esta tecnologia.

1.4 Organização deste documento

Ao longo deste trabalho, estudaremos o que é o protocolo Solid, procurando explorar algumas aplicações que já fazem uso do mesmo em 2023 com a intenção de mapear e estudá-las para aprender o quanto da tecnologia foi adotada pela comunidade. Além disso, foram desenvolvidas duas aplicações demonstrativas voltadas para o público-alvo do *smartUnB.ECOS* a fim de validar se a adoção do Solid é uma boa solução para o ecossistema educacional e quais possibilidades e desafios esta adoção traria ao mesmo. No Capítulo 2 explicamos melhor o que é o ecossistema *smartUnB.ECOS*, seguido dos atuais problemas com dados educacionais, antes de explicar o que é e como funciona a plataforma

Solid. No Capítulo 3 listamos e apresentamos algumas aplicações que implementam Solid. No Capítulo 4 descrevemos o processo de elaboração e desenvolvimento de dois aplicativos que fazem uso do Solid, com objetivo de demonstrar a interoperabilidade proveniente do protocolo Solid e provar sua utilidade para o *smartUnB.ECOS*. No Capítulo 5 apresentamos os resultados e aprendizados adquiridos a partir do desenvolvimento dessas aplicações. No Capítulo 6 concluímos o presente trabalho com uma visão geral sobre a adoção do Solid e sobre a experiência de desenvolvimento com essa tecnologia.

Capítulo 2

Fundamentação

2.1 Do lado do problema: Privacidade de dados educacionais

A coleta e análise de dados por sistemas vêm ganhando notoriedade nos últimos anos, ao ponto de governos elaborarem leis e estatutos entorno da privacidade de dados, como a *Family Educational Rights and Privacy Act (FERPA)*, *General Data Protection Regulation (GDPR)* e LGPD [8, 17]; e grandes nomes da tecnologia alterando seus termos e políticas de privacidade para entrar em conformidade com elas. Comumente os sítios Web pedem autorização para gerenciar *cookies* [3]: isto é fruto de tais leis.

Essa importância alocada aos dados não é descabida. Existem estudos sobre desenvolvimentos de algoritmos e ferramentas para pesquisa e análise de dados gerados por usuários em redes sociais com intuito de melhor direcionar propagandas relevantes a eles [18, 19], baseados em postagens voluntariamente servidas a essas redes sociais por seus usuários, sem que muitas vezes eles nem tenham total compreensão de como seus dados são processados [20]. Tais algoritmos podem ser bem poderosos, e até intrusivos, como atesta o pai de uma adolescente dos Estados Unidos, em uma reportagem que gerou um grande impacto na época [21], na qual uma empresa desenvolveu um programa de análise de compras para sugerir cupons de desconto. Este programa, ao analisar padrões de compra de uma adolescente, enviou cupons com descontos para itens de bebê, insinuando que ela estaria grávida. O pai, ao descobrir esses cupons, foi reclamar com o gerente desta loja, acusando que a loja estaria estimulando a filha a engravidar, sem saber que ela já estava de fato grávida, como o algoritmo corretamente supôs.

O uso de dados no ambiente educacional também podem ser aplicados para contribuir com melhorias de *softwares* e ferramentas educacionais [22], como analisando como estudantes usam certo *software* para resolver problemas, quanto tempo levam, com qual

facilidade interagem com o *software* e, a partir da análise destes dados, professores, aluno e membros da comunidade educacional podem sugerir mudanças para melhorar o aprendizado. A questão, então, gira mais uma vez entorno da privacidade, em como esses ambientes educacionais coletam estes dados, e quais dados são esses. Existem *Massively Open Online Courses (MOOCs)* que coletam cerca de 20GB de dados de usuários por curso sobre quanto tempo passam no curso, com que frequência entram, qual escolaridade têm, entre outros muitos dados, que são usados [8] para melhor direcionar *marketing* de outros cursos dentro do próprio ambiente educacional para os alunos. No entanto, esses dados também poderiam ser compartilhados, por exemplo, com editoras de livros acadêmicos para desenvolver melhores exercícios ou estruturar melhor seus capítulos [8], sem que os alunos usuários sequer façam ideia que seus dados estão sendo analisados para melhor entender sua rotina de estudos e ajudar empresas terceiras a melhorarem seus produtos e aumentar os lucros como consequência.

Isso é levando em conta apenas o lado da privacidade e falta de controle por parte dos usuários sobre os dados coletados. Ainda existe o lado da redundância de dados, no qual para todo *software* ou ferramenta educacional que um usuário optar por usar, ele informará seu nome, sua idade, sua escolaridade, seu curso, cidade, entre outros; e o lado da dependência às ferramentas e softwares que usam há mais tempo, justamente por terem mais dados de algum usuário em questão que fazem com que concorrentes não consigam competir para oferecer um serviço que tenha tanto apelo àquele usuário em questão.

O problema é claro: do lado dos alunos e usuários destes ambientes educacionais no geral, eles não têm total acesso a seus dados, não sabem o quanto eles são usados e com quais finalidades, devem repetir e fornecer novamente dados já informados anteriormente em outras plataformas, e não têm como saber quem mais tem acesso a eles. Do lado dos provedores destes serviços se mostra uma falta de oportunidade de concorrer de forma justa com um acervo massivo de dados de usuários sob controle de um produto específico. Felizmente, também existe uma solução clara, proposta pelo “pai da Web”, Tim Berners-Lee.

2.2 Do lado da solução proposta: Solid, por uma Web descentralizada

Solid é uma especificação de projeto descentralizado para aplicações sociais na Web [9, 23] (como a *smartUnB.ECOS*). Nela, os dados dos usuários são geridos de forma independente das aplicações que as consomem, ou seja, o usuário mantém o controle de seus dados e gerencia quais aplicações têm acesso a quais dados, pois cada usuário armazena seus dados

em um centro de dados particular *online* chamados *Personal Online Datastore (POD)*, e aplicações fazem requisições a esses PODs com autorização explícita do usuário.

A adoção do Solid, não só no âmbito educacional, mas para toda a Web é, além de trazer a autonomia de seus dados para o usuário, também a descentralização do monopólio de grandes empresas através de dados, “é sobre criar mercados”, como diz Berners-Lee [24]. A tecnologia busca ser a solução de três desafios propostos por Tim Berners-Lee para a Web do futuro, visando trazer o controle dos dados de volta para os usuários, impedir a distribuição de desinformação e trazer mais transparência em propagandas políticas *online* [25].

Isso é possível, pois com o Solid, um usuário pode armazenar todos os seus dados apenas uma vez em seu POD (combatendo a redundância de dados), saber exatamente quais dados serão acessados e permitir acesso a dados específicos para aplicações específicas, como visto na Figura 2.1¹, fornecendo um campo justo para que aplicações concorrentes possam fazer serviços em cima do mesmo grupo de dados, sendo benéfico para o usuário final que poderá escolher o melhor sem se sentir preso por ter seus dados vinculados a uma aplicação (por exemplo, bibliotecas de músicas que um usuário construiu ao longo de anos em uma aplicação específica poderiam ser compartilhadas com outra aplicação apenas com uma rápida autorização de acesso do usuário). Além disso, Solid já está de acordo com as leis de proteção de dados como a GDPR e LGPD por sua própria definição; os dados do usuário estão sob controle do próprio usuário.

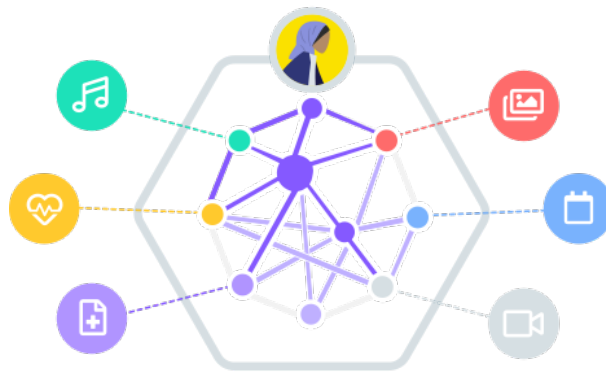


Figura 2.1: Diagrama de Descentralização do Solid¹.

O Solid funciona com um usuário guardando seus dados em PODs, aos quais ele tem total acesso e liberdade para escolher quais aplicações podem acessá-los. As aplicações

¹Imagem retirada da página oficial do projeto Solid: <https://solidproject.org/about>

Web utilizam protocolos de autenticação para descobrir o perfil do usuário e caminhos para o POD específico do usuário que contém os dados daquela aplicação. Para garantir a segurança e consistência dos dados acessados, um provedor de indentificação (WebID) [26] gerencia a autenticação e provê o documento que contém as informações criptografadas do usuário para a aplicação em questão. Os dados das aplicações são armazenados nos PODs, que por sua vez são armazenados em POD-servers.

O acesso a estes dados é feito com requisições REST e suas respostas são “*application agnostic*”, ou seja, independe da aplicação sendo desenvolvida, os dados sempre poderão ser consumidos sem ser necessário fazer alterações nos servidores [23]. A comunicação entre PODs e aplicações ou navegadores é possível devido a um padrão de conexão entre dados da Web conhecido como *Linked Data* [27]. O Solid utiliza este padrão para garantir o compartilhamento de dados de forma descentralizada, padronizando o vocabulário de dados armazenados em PODs diferentes. Ele utiliza *Uniform Resource Identifier (URI)* para identificar recursos estruturados *Resource Description Framework (RDF)* [28]. Dentro da Web-semântica [29] existe ainda o conceito de vocabulários [30], que podem ser entendidos como RDFs para entidades específicas, por exemplo *vCard* para referenciar informações sobre o usuário, como endereço e nome [31]; ou o *FOAF - Friend of a Friend* que é utilizado para descrever relacionamentos, como pessoas que o usuário conhece [32].

Respeitar estes padrões permite que aplicações independentes consigam trocar informações sem compartilharem dados diretamente entre si e resulta, fora a privacidade e controle dos dados, na característica mais importante do Solid: a interoperabilidade. Este estudo se propõe a avaliar como seria implementar aplicações que façam uso do Solid ao ecossistema *smartUnB.ECOS*.

2.3 smartUnB.ECOS: aprendendo sobre o ecossistema de aprendizagem e ensino

Contando com esta visão atual da tecnologia de educação com ferramentas e sistemas de *e-learning* voltados para o ensino que vimos anteriormente, a professora doutora e orientadora deste trabalho, Germana Menezes da Nóbrega, idealizou o *smartUnB.ECOS*, um ecossistema de aprendizagem e ensino, ou seja, um conjunto de sistemas e ferramentas de *e-learning*, desenvolvido para apoiar os alunos e alunas do Departamento de Ciência da Computação (CIC). Esse apoio aos alunos é dado em duas vias: a primeira na forma de oportunidade de aprendizado no sentido de realizar pesquisa, projeto e implementação de componentes que enriquecem o ecossistema; e a segunda na forma de retorno à comunidade do departamento de projetos e sistemas úteis para o uso e aprendizado do corpo discente. O presente estudo faz proveito desta primeira via de aprendizagem.

Atualmente, o *smartUnB.ECOS* é composto por 9 elementos que estimulam uma variedade dinâmica de conexão entre aprendizagem formal e informal [1]. Esses elementos vão de portais de acesso ao ecossistema, redes sociais, ferramentas e sistemas de aprendizagem, entre outros, que podem ser observadas na Figura 2.2. Vamos descrever brevemente cada elemento que compõe esse ecossistema a título de completude e de prover maior entendimento sobre o ecossistema em questão ao qual este trabalho contribui.

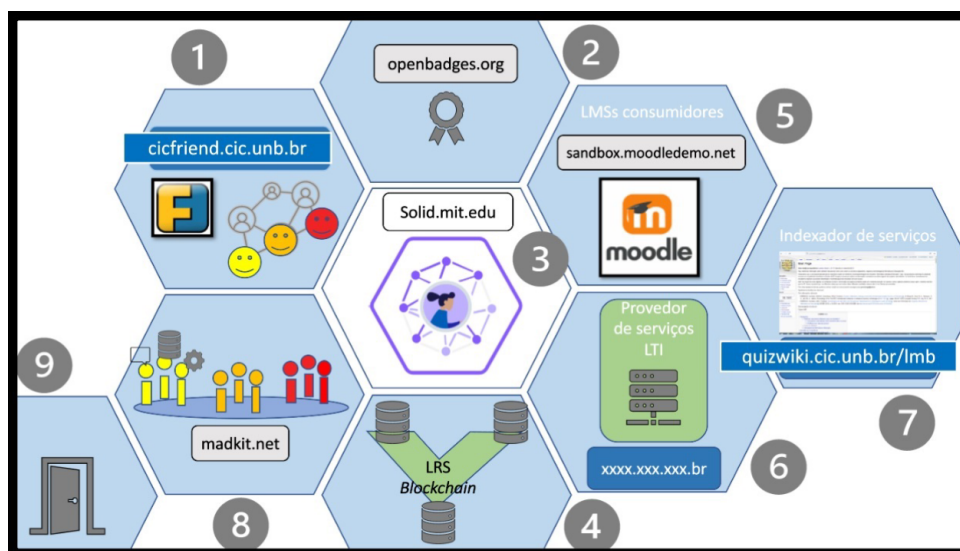


Figura 2.2: Arquitetura 2021 smartUnB.ECOS [1].

1. Rede Social Descentralizada: plataforma de rede social voltada para a comunidade do CIC, baseada no *Friendica*²;
2. Plataforma de gestão de *Badges* Digitais: plataforma para gerenciar a criação e atribuição de medalhas digitais para o corpo discente do CIC por mérito em atividades informais (gamificação acadêmica);
3. *POD-Server*: implementação e manutenção de um provedor POD dedicado a abrigar PODs dos usuários da comunidade do CIC;
4. *Learning Records Store (LRS)* em *blockchain*: armazenamento de forma distribuída de registros de aprendizagem de alunos para fins analíticos de aprendizagem, com autorização do usuário;
5. *Learning Management Systems (LMS)* com suporte a *Learning Tools Interoperability (LTI)*: são plataformas de gestão de aprendizagem tradicionais (como *Moodle*), porém fazendo uso de LTIs, que fornecem suporte a disciplinas de cursos ofertados;

²<https://friendi.ca>

6. Provedor de serviços LTI: provedor de ferramentas, serviços dedicados ou específicos para utilização em disciplinas, que complementam LMSs na prestação de seus serviços;
7. Repositório de Recursos Educacionais: ambiente de indexação semântica para os serviços dedicados do provedor LTI e também para recursos da produção nacional e internacional de relevância para a comunidade do CIC;
8. Sociedade de Assistentes Artificiais: agentes autônomos que cooperam em prol da saúde acadêmica do corpo discente, seguindo o meta-modelo *Agent/Group/Role (AGR)*;
9. *Learning Companion*: interface do ecossistema, provê o acesso aos demais elementos e personalização do usuário.

Para este trabalho, o elemento relevante é a implementação de um *POD-Server* para prover os PODs aos alunos e funcionários do CIC, o que facilitará e estimulará o uso de aplicações Solid-compatíveis, como as desenvolvidas neste trabalho. Existem outros trabalhos em desenvolvimento sendo realizados em volta da implementação desse *POD-Server*, bem como em outras áreas do ecossistema, caso o leitor tenha interesse de saber mais [1].

2.4 Tecnologias Utilizadas

Para o desenvolvimento das aplicações, foram utilizadas diversas ferramentas e tecnologias. Vamos listar cada uma e comentar sobre suas contribuições para este trabalho.

Notion

Utilizamos Notion³ para organizar o passo-a-passo do desenvolvimento das aplicações, listando as várias etapas para conseguir dividir grandes funcionalidades em pequenas tarefas e implementar uma de cada vez sem perder o escopo maior do projeto.

Figma

Para a elaboração do *design*, o Figma⁴ foi escolhido para auxiliar no desenvolvimento da UI das aplicações, por ser fácil de utilizar e permitir testar as melhores formas de organizar as informações na tela.

³<https://www.notion.so/>

⁴<https://www.figma.com/>

Visual Studio Code

O editor⁵ de texto mais popular [33] para desenvolvimento também foi utilizado neste trabalho. A familiaridade com ele foi o que mais contribuiu para essa escolha.

Git e Github

Utilizamos Git para o versionamento do código e hospedamos os repositórios no *site* do Github⁶. Novamente, a escolha foi feita devida a familiaridade com as ferramentas.

Node Package Manager (NPM)

Para executar as bibliotecas em JavaScript necessárias, um gerenciador de pacotes é recomendado. A escolha do NPM⁷ foi feita por causa da biblioteca que facilita o trabalho com Solid exigir uma versão específica do NPM: LTS-16.0.0. Esta é uma versão velha (a versão LTS mais recente é a 20), e traz consigo uma vulnerabilidade grave acusada pelo *console*.

ReactJS

Uma biblioteca de JavaScript para desenvolvimento *frontend*, utilizamos o React⁸ para implementar a UI das aplicações. O uso de *hooks* e *states* possibilitou criar aplicações interativas com mais facilidade se comparado à utilização de JavaScript convencional.

Iniciamos o projeto com *create-react-app* para ter acesso a uma estrutura inicial do projeto. Definimos a seguir um padrão de projeto que consiste em separar componentes em lógica, JSX e estilo; desta forma, a estrutura de pastas utilizada em ambos projetos foi a seguinte:

- **src**: pasta raiz do projeto
 - **api**
 - * **firebase**: contém arquivos de configuração e funções do Firebase (somente RUview)
 - * **functions**: contém os arquivos com as funções de uso com Solid
 - **assets**
 - * **images**: contém as imagens do projeto

⁵<https://code.visualstudio.com/>

⁶<https://github.com/>

⁷<https://www.npmjs.com/>

⁸<https://react.dev/>

- **components:** contém as pastas de cada componente do projeto. Cada componente é composto por um *index* responsável pela lógica (states, funções), um *Display.jsx* para exibir o componente, e um *styles* que contém os estilos
- **hooks:** contém os hooks personalizados do projeto. Foram utilizados hooks para sincronizar os states globais com os dados buscados do POD do usuário
- **pages:** contém as pastas de cada página da aplicação. Funcionam igual aos componentes
- **util:** contém arquivos facilitadores do desenvolvimento do projeto, como arquivo com os códigos de cores HEX e funções de tratamento de texto
- **index.css:** estilos globais
- **index.js:** ponto de entrada da aplicação. É onde os hooks e a biblioteca de rotas são chamados

Firestore

O Firestore⁹ é uma coleção de serviços para armazenamento e hospedagem de dados *online*. Dessa coleção, fizemos uso do *Firestore*, um SGBD em nuvem, e do *FirestoreAuth*, um serviço que provê autenticação de maneira fácil e segura. Utilizamos, também, a biblioteca JavaScript do Firestore que traz funções para comunicar sua aplicação com os serviços. O Firestore foi utilizado somente no *RUview* para autenticar usuários administradores e armazenar os cardápios e refeições gerenciadas por esses administradores. O acesso a essas funcionalidades na aplicação está na rota */admin* que precisa ser acessada manualmente pela URL do navegador, e em seguida autenticada com o *FirestoreAuth*.

PodPro

O PodPro¹⁰ foi de grande ajuda para o desenvolvimento das aplicações deste trabalho, pois permitiu abrir todos os PODs e visualizar os conteúdos para garantir que estavam sendo salvos corretamente, além de também permitir editar esses conteúdos para realizar diferentes testes dentro das aplicações e garantir o funcionamento correto.

Vercel

Vercel¹¹ é uma plataforma para hospedar *sites* de forma gratuita e fácil, bastando vincular com o repositório armazenado no Github. Utilizamos ela para subir ambas aplicações.

⁹<https://firebase.google.com/>

¹⁰<https://podpro.dev/>

¹¹<https://vercel.com/>

Capítulo 3

Aplicações em Solid

Com o objetivo de juntar uma coleção de aplicações que servirão de referência e inspiração para futuras aplicações Solid, pesquisamos projetos e aplicações que fazem uso desta tecnologia hoje, e as exploramos neste capítulo. Procuramos reunir conhecimento sobre os avanços com a tecnologia Solid em outros países do mundo, bem como avaliar como está sendo a adoção desta tecnologia nos últimos anos. Para cada aplicação listada, procuramos entender os seguintes pontos:

- O que faz a aplicação?
- Como implementa o Solid?
- Em qual país foi desenvolvido?
- Em que ano?
- Tem objetivo comercial ou não?

Com essas respostas, poderemos ter uma ideia melhor sobre qual estágio de adoção esta tecnologia está perante a comunidade mais ampla. No Capítulo 4 e 5 veremos também como Solid se apresenta para a comunidade de desenvolvimento.

3.1 Provedor POD

Todas as aplicações que fazem uso do Solid pedem, em algum momento, para ter acesso a algum POD. No caso do *smartUnB.ECOS*, o aluno ou professor terá acesso a um POD personalizado hospedado dentro do próprio ecossistema; entretanto, esta etapa ainda não foi concluída até o momento de escrita deste trabalho e, para poder testar as aplicações selecionadas, o usuário deve informar algum POD para que a aplicação funcione.

Criar e hospedar seu próprio POD é uma alternativa prevista pelo projeto Solid [9]. No entanto, diversos provedores de PODs já existem e muitas dessas aplicações já possuem apoio para alguns desses provedores. Por isso, enquanto não temos acesso ao POD do ecossistema, optamos por utilizar um POD do provedor *inrupt.net*. Outros provedores foram testados também, mas este se mostrou o mais adequado para utilizar com as aplicações devido à facilidade de uso, tanto com as aplicações quanto com o gerenciamento do próprio POD depois de criado pela sua interface de navegador padrão.

3.2 Aplicações

A busca e escolha por essas aplicações e projetos ocorreu por algumas fontes: no *site* oficial do projeto Solid, na área de aplicações destacadas¹; em postagens nos fóruns Reddit²; em testemunhos nos *sites* de provedores PODs (listados no *site* oficial do projeto Solid); em projetos noticiados na mídia; e em *sites* parceiros de outros projetos que usam o Solid. Ao todo, selecionamos 19 aplicações. Dentre elas, conseguimos utilizar, testar e estudar o código fonte de 9 projetos. O restante dos projetos selecionados ou eram pagos ou eram proprietários, impossibilitando o estudo de código fonte.

A seguir vamos apresentar cada projeto, respondendo as questões levantadas no início deste capítulo.

3.2.1 Curso de Arquitetura de Software da Universidade de Oviedo

Desde 2019, a disciplina de Arquitetura de Software³, ministrada para o sexto semestre do Curso de Engenharia de Software da Universidade de Oviedo, Espanha, traz o desenvolvimento e publicação de aplicações Solid-compatíveis de forma *open-source* e devidamente documentado. Sob as orientações e ensinamentos do professor responsável pela disciplina, Jose Emilio Labra Gayo, os alunos se dividem em grupos e desenvolvem uma aplicação Solid por ano. Os melhores trabalhos de cada ano são destacados e enviados para concorrer no “desafio Solid”, onde podem aparecer destacados na página oficial do *site* do projeto Solid.

Todas as aplicações desenvolvidas são *open-source* e com licença MIT. Os projetos são documentados semana-a-semana pelos próprios alunos e a página da disciplina disponibiliza tanto projetos em Inglês quanto em Espanhol. São aplicações com viés não-comercial,

¹<https://solidproject.org/apps>

²https://www.reddit.com/r/SOLID/comments/pxub34/useful_resources_for_solid/

³<https://arquisoft.github.io/>

servindo apenas como demonstração dos tipos de projetos que já são possíveis de serem feitos com Solid hoje.

DeChat

O objetivo do DeChat é criar uma aplicação descentralizada de conversa entre usuários de PODs distintos, completos com listagem de amigos e edição de perfil. Esse foi o primeiro projeto a ser publicado pela disciplina e foi destacado na página oficial do projeto Solid em 2019.

ViaDe

Viade é a segunda aplicação do Curso de Arquitetura de Software da Universidade de Ovideo. Ela consiste em elaborar e armazenar rotas no mapa e salvá-las no POD do usuário. Projeto de 2020.

Radarín

Em 2021 os alunos publicaram o Radarín, uma aplicação de geolocalização similar ao foursquare, armazenando os lugares visitados no POD do usuário.

DeDe

Dede foi a aplicação de 2022. Consiste em utilizar o endereço registrado no POD do usuário para calcular o frete na entrega de um produto do *site* de *e-commerce* desenvolvido. Os produtos e estoque são armazenados em um banco de dados centralizado.

Lomap

Lomap é a aplicação mais recente do Curso durante a escrita deste trabalho, publicada no ano de 2023. Desta vez o projeto repetiu o objetivo e se assemelha ao ViaDe no sentido de armazenar rotas e trajetos no POD do usuário para amigos verem e comentar. O diferencial nesta versão é a implementação de um sistema de gamificação em volta do compartilhamento de rotas.

3.2.2 Media Kraken

Este aplicativo é exibido como aplicativo destaque na página do projeto Solid⁴. Ele consiste em indicar filmes que o usuário já viu e avaliá-los, armazenando todos esses dados nos PODs de cada usuário. Desenvolvido em 2020 por Noel De Martin, em Barcelona,

⁴<https://noeldemartin.github.io/media-kraken/login>

Espanha, esse aplicativo é mais um com foco principal na demonstração do uso do Solid e sem viés comercial.

3.2.3 Notepod

Notepod⁵ é uma simples aplicação de um CRUD de notas que são salvas no POD do usuário autenticado. Essa aplicação foi criada individualmente pelo holandês Vincent Tunru em 2019, e é mais uma aplicação de demonstração não-comercial de código aberto.

3.2.4 PodPro

PodPro⁶ é uma ferramenta para visualizar todos os recursos de um POD de maneira limpa e de fácil leitura. Desenvolvida por Jasmine Leonard em 2022, no Reino Unido, essa ferramenta foi de grande ajuda para o desenvolvimento das aplicações no Capítulo 4 deste trabalho. Com ela, é possível abrir e ler o que tem dentro de cada POD, podendo modificar, acrescentar ou excluir qualquer recurso de maneira bem simples e objetiva. Essa é também a primeira aplicação selecionada que julgamos possuir um viés comercial, apesar de grátis. O intuito dela é servir como principal ferramenta de auxílio ao desenvolvimento com PODs.

3.2.5 Sigmafied

Sigmafied⁷ é outro projeto individual, desenvolvido pela australiana Sharon Stratsianis para implementar uma aplicação Web utilizando Solid e React (uma biblioteca Javascript popular). A aplicação consiste em identificar a localização atual do usuário e registrar em seu POD, similar a Radarín do curso de Oviedo. Publicada em 2019.

3.2.6 graphMetrix

Aplicação que permite com que o usuário possa controlar e administrar seu POD, podendo ver estatísticas e grafos sobre várias informações sobre seu POD, como controle de acesso e gerenciamento de arquivos⁸. Empresa localizada nos Estados Unidos (São Francisco) e em atuação desde 2018. Também possui um viés comercial e uma proposta igual a do PodPro visto anteriormente, porém esse serviço é pago.

⁵<https://notepod.vincenttunru.com/>

⁶<https://podpro.dev/>

⁷<https://sigmafied.com/Welcome>

⁸<https://graphmetrix.net/>

3.2.7 Pojeto do sistema de saúde britânico - NHS

Em 2021, o Serviço Nacional de Saúde do Reino Unido, em conjunto com a empresa Janeiro Digital⁹, revelaram seu trabalho na implementação de um sistema de saúde descentralizado, onde os pacientes estão tendo acesso à PODs individuais nos quais são armazenados todos os dados relevantes à saúde daquele paciente. Este foi um dos projetos que mais chamou atenção da comunidade e que conseguiu trazer certa visibilidade para o Solid.

3.2.8 POD Servers do governo de Flandres

O governo de Flandres, Bélgica, distribuiu 6 milhões de PODs em 2022 para estudantes para que eles pudessem armazenar conteúdos digitais diversos¹⁰. O objetivo é estimular o desenvolvimento de aplicações descentralizadas, uma vez que estas já teriam um público e um conjuntos de dados para fazer com que aplicações Solid-compatíveis cresçam.

3.2.9 My PDS - PODs da BBC com diferentes perfis

A BBC também está oferecendo PODs para que os usuários possam armazenar suas preferências a respeito do conteúdo que consomem em aplicações de entretenimento como *Spotify*, *Netflix*, *Disney+*, etc, saúde, financeiro e social¹¹. O serviço chama My PDS e ainda está em desenvolvimento, sem previsão de lançamento. Apesar disso, este projeto foi escolhido, pois potencialmente mostra a adoção do Solid por uma empresa grande como a BBC e, caso obtenha sucesso, poderá impulsionar a tecnologia.

3.2.10 karamel

Karamel¹² é uma empresa da Alemanha voltada a busca de empregos de forma descentralizada. A proposta é armazenar currículos e capacitações técnicas nos PODs dos usuários e direcionar empresas contratantes a estes usuários. Viés comercial, em atuação desde 2022.

3.2.11 UZ BRUSSEL RE-USE MEDICAL

Empresa Belga de reuso descentralizados de dados médicos¹³. A proposta é armazenar dados médicos em PODs para possibilitar o compartilhamento de exames e outras in-

⁹<https://www.janeirodigital.com/blog/janeiro-digital-at-solid-world-nhs-personal-health-stores-wi>

¹⁰<https://www.inrupt.com/blog/digital-flanders-reconnects-citizens-with-their-data-through-inrupts>

¹¹<https://diginomica.com/shifting-power-balance-bbc-wants-give-individuals-control-their-data-onli>

¹²<https://karamel.career/over-ons/>

¹³<https://get.use.id/health>

formações de saúde entre médicos, pacientes e aplicativos de saúde. Viés comercial, em funcionamento desde 2022.

3.2.12 Linckr

Linckr¹⁴ processa e armazena os dados pessoais de identificação de cada usuário em seus PODs, como CNH e título de eleitor, e em seguida faz um intermédio entre o usuário e diversos serviços que exigem identificação, como compra de casas e declaração de impostos de renda. Empresa alemã em funcionamento desde 2022.

3.2.13 Konsolidate

Empresa de desenvolvedores com foco em aplicações Solid-compatíveis¹⁵. Desde 2021, essa empresa belga faz aplicações e desenvolve projetos descentralizados.

3.2.14 VITO (BIBOPP)

Seguimento de uma empresa holandesa que em 2022 iniciou o desenvolvimento de um serviço que analisa dados médicos presentes em POD de pacientes para identificar problemas de saúde com antecedência e tratá-los¹⁶.

3.2.15 We Are

We Are¹⁷ é uma provedora de PODs da Bélgica focados em lidar com dados de saúde. Desde 2021, eles proveêm PODs para algumas empresas de saúde, inclusive a VITO vista anteriormente.

Reunindo as aplicações em uma visão geral, temos a Tabela 3.1, na qual podemos observar o ano, país e viés comercial de cada projeto. Comentaremos esses resultados e o que eles nos dizem sobre a adoção do Solid no Capítulo 5 mais adiante. A seguir iremos descrever o desenvolvimento das aplicações em Solid com foco educacional propostas neste trabalho.

¹⁴<https://www.linckr.com/>

¹⁵<https://www.konsolidate.eu/>

¹⁶<https://vito.be/en/news/live-your-healthiest-life-thanks-kempen-pilot-project-bibopp>

¹⁷<https://we-are-health.be/en>

Tabela 3.1: Aplicações Solid.

Aplicação	País	Ano	Comercial?
DeChat	Espanha	2019	Não
ViaDe	Espanha	2020	Não
Radarín	Espanha	2021	Não
DeDe	Espanha	2022	Não
Lomap	Espanha	2023	Não
Media Kraken	Espanha	2020	Não
Notepod	Holanda	2019	Não
PodPro	Reino Unido	2022	Sim
Sigmafied	Austrália	2019	Não
graphMetrix	Estados Unidos	2018	Sim
NHS britânico	Reino Unido	2021	Sim
PODs governo Flandres	Bélgica	2022	Sim
My PDS	Reino Unido	-	Sim
karamel	Alemanha	2022	Sim
UZ Brussel re-use	Bélgica	2022	Sim
Linckr	Alemanha	2022	Sim
Vito (BIBOPP)	Holanda	2022	Sim
We Are	Bélgica	2021	Sim

Capítulo 4

Desenvolvimento dos aplicativos

O desenvolvimento de aplicações Solid é uma necessidade para que a tecnologia ganhe mais notoriedade. Com o estudo das aplicações listadas no capítulo anterior servindo de inspiração para o desenvolvimento das aplicações de demonstração para o *smartUnB.ECOS* que veremos neste capítulo, ele mostra o que já é possível de ser realizado com a tecnologia atualmente. No capítulo anterior pudemos ver um pouco sobre a adoção do Solid pela comunidade mundial; neste capítulo veremos sobre a experiência de desenvolver com essa tecnologia.

Sendo assim, propomos a criação e implementação de duas aplicações Solid-compatíveis com temáticas voltadas para o contexto educacional, ou seja, que fariam sentido em serem utilizadas frequentemente por alunos e professores de escolas e universidades. O objetivo por trás do desenvolvimento desses aplicativos é mostrar benefícios de adotar o Solid e desenvolver aplicações com esta tecnologia para compor o *smartUnb.ECOS*. Para mostrar isso, faremos uso da propriedade de interoperabilidade do Solid, como vimos no Capítulo 2, no desenvolvimento do *RUview* e do *Tutor*, as aplicações de demonstração propostas.

4.1 Descrição das aplicações

O *RUview* permite que os alunos possam visualizar o cardápio da semana e quais são as refeições do RU disponíveis no dia do acesso à plataforma. Caso autorizem o acesso a seus PODs, estes alunos podem ainda avaliar estas refeições caso gostem ou desgostem delas, e também ver se o que seus amigos pensam sobre as mesmas refeições. Existe também um sistema simples de edição de amigos em seu perfil Solid, e ainda uma área para administradores, onde é possível criar e modificar as refeições presentes no sistema e atualizar o cardápio com elas. Esta última parte não é descentralizada, com as informações das refeições, cardápios e de autenticação armazenadas no Firebase, um SGBD hospedado em nuvem. A Figura 4.1 mostra um visão geral do funcionamento do *RUview*. O usuário

autoriza a aplicação a acessar seu POD e ter acesso ao seu WebID. Uma vez autenticado, a aplicação busca as refeições e cardápios do Firebase, busca os JSONs de refeições curtidos e descurtidos do POD do usuário, e acessa a lista de amigos do usuário. Para cada amigo nessa lista, a aplicação busca os JSONs das refeições daquele amigo para poder exibir quais amigos gostaram ou não da refeição daquele dia. Toda interação de curtida e descurtida das refeições alteram os JSONs correspondente no POD do usuário. Os JSONs de refeições curtidas (idem para descurtidas) são *arrays* que contêm objetos com a seguinte estrutura:

```

1  {
2      id_refeicao: informa o identificador da refeicao no Firebase,
3      dia_semana: informa o dia da semana que aquele prato apareceu no
                    cardapio,
4      hora_refeicao: informa se a refeicao e o cafe da manha, almoco
                    ou janta
5  }

```

Listagem de Código 4.1: Estrutura do JSON da refeição

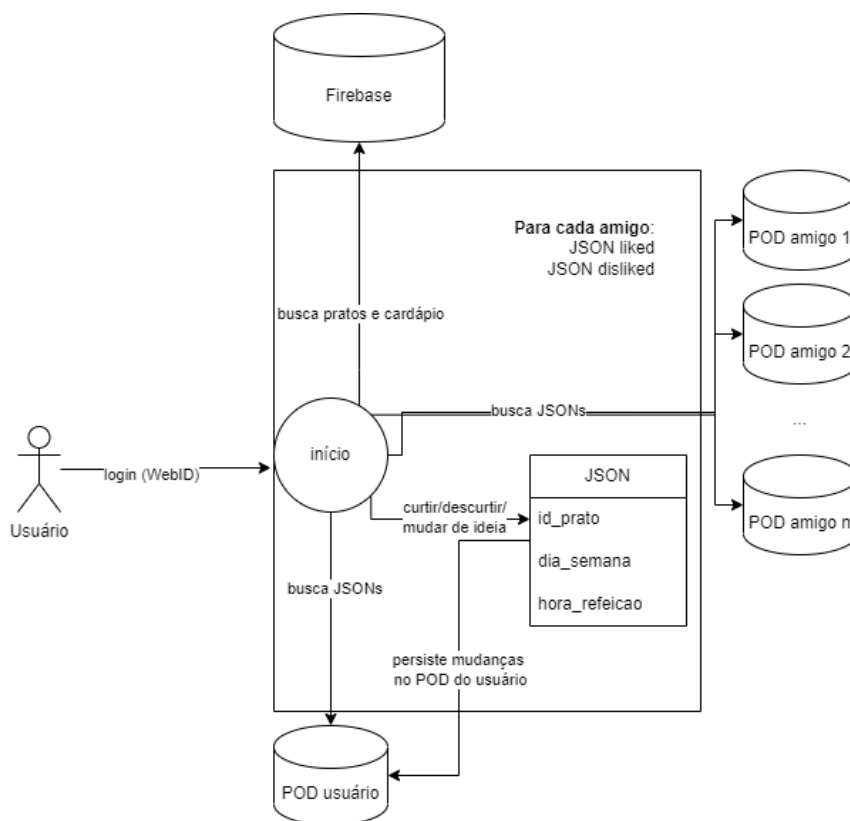


Figura 4.1: Diagrama Funcionamento RUview.

Tabela 4.1: STATUS dias da agenda.

Código	Disponibilidade
0	Indisponível
1	Livre
2	Pendente
3	Ocupado

Já o *Tutor* consiste em facilitar o agendamento de reuniões de estudos entre dois alunos. Uma vez que o aluno autoriza a aplicação a acessar seu POD, ele pode criar sua agenda com seus horários livres da semana e também ver as agendas de seus amigos para enviar propostas de reunião para cada um em seus horários livres. A Figura 4.2 mostra um visão geral do funcionamento do *Tutor*. O usuário autoriza a aplicação a acessar seu POD e ter acesso ao seu WebID. Uma vez autenticado, a aplicação busca os JSONs da agenda e dos compromissos do POD do usuário, e acessa a lista de amigos do usuário. Para cada amigo nessa lista, a aplicação busca os JSONs da agenda, dos compromissos e dos pratos curtidos pelo *RUview* daquele amigo para poder exibir as agendas e compromissos atualizados. O JSON dos pratos curtidos é utilizado para comparar se existe algum prato na lista com o dia atual e, se nesse caso, existe algum prato com o horário que coincide em um horário livre na agenda do amigo e do usuário; se este for o caso, a aplicação exibe uma mensagem para sugerir este horário para reunião. Toda interação de criação, remoção ou atualização dos compromissos ou agenda alteram os JSONs correspondente no POD do usuário. O JSON da agenda tem a seguinte estrutura:

```

1   {
2     updated_at: informa quando foi modificada por ultimo,
3     [dia_hora]: informa o status de disponibilidade daquele dia e
                horario. Os dias variam de segunda a domingo e as horas de 06
                h ate 22h, em intervalos de 2 em 2 (63 entradas no total)
4   };

```

Listagem de Código 4.2: Estrutura do JSON da agenda

Os *status* possíveis de disponibilidade de cada dia e hora da agenda pode ser visto na Tabela 4.1.

O JSON de compromissos é um *array* que contém objetos com a seguinte estrutura:

```

1   {
2     id: identifica esse compromisso,
3     dia_hora_reuniao: informa o dia e a hora da reuniao na agenda,
4     updated_at: informa quando foi modificada por ultimo,
5     webID_amigo: informa com qual amigo combinou o compromisso,
6     status: informa qual situacao desse compromisso };

```

Listagem de Código 4.3: Estrutura do JSON do compromisso

Tabela 4.2: STATUS de compromissos.

Código	Situação
0	Pendente
1	Confirmado
2	Cancelado por você
3	Cancelado pelo amigo
4	Aguardando sua resposta

Os *status* possíveis de cada compromisso pode ser visto na Tabela 4.2.

Cada aluno pode aceitar ou recusar propostas de reuniões. Essas estruturas descritas podem ser observadas na Figura 4.3. Os específicos de como essas aplicações implementam o Solid serão exibidos mais a frente neste capítulo.

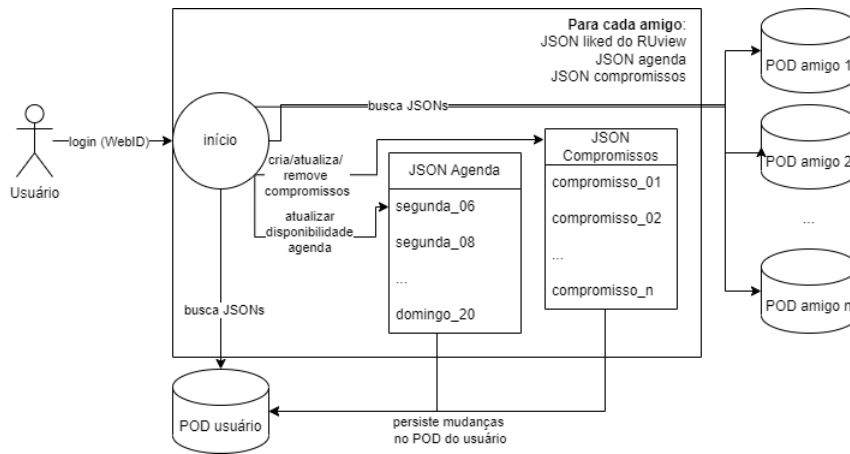


Figura 4.2: Diagrama Funcionamento Tutor.

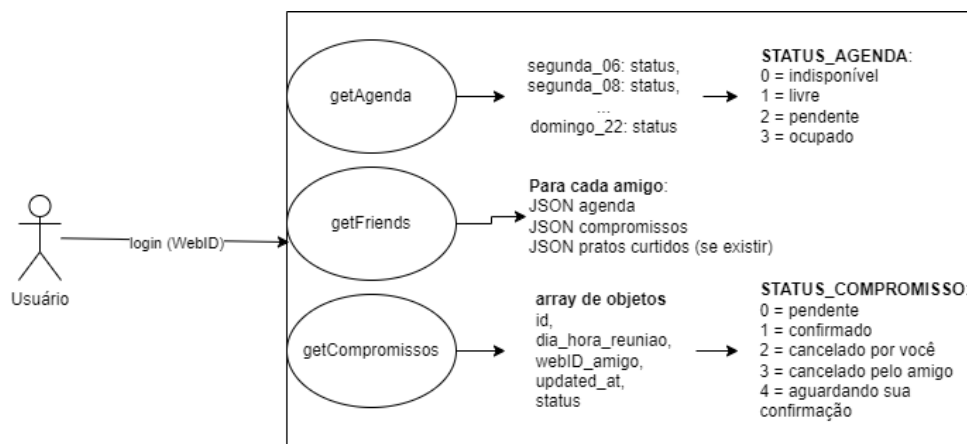


Figura 4.3: Diagrama da Estrutura das principais funções do Tutor.

A interoperabilidade se mostra presente no funcionamento do *Tutor*, que procura por dados criados pelo *RUview* nos PODs dos amigos e do próprio aluno para sugerir possíveis oportunidades de reunião, com a ideia por trás sendo se ambos gostam daquela refeição, então os dois estarão pela Universidade naquele horário, sendo uma boa oportunidade para se reunirem. Isso é uma suposição ingênua, é claro, mas é suficiente para mostrar um possível benefício para o *smartUnB.ECOS*. Sabendo como uma aplicação guarda os dados nos PODs dos usuários, torna-se possível utilizá-los para o funcionamento de outras aplicações. Esse comportamento pode ser observado na Figura 4.4, que exemplifica duas aplicações acessando o POD do usuário e a segunda aplicação exibindo dados criados tanto por ela própria quanto pela primeira aplicação.

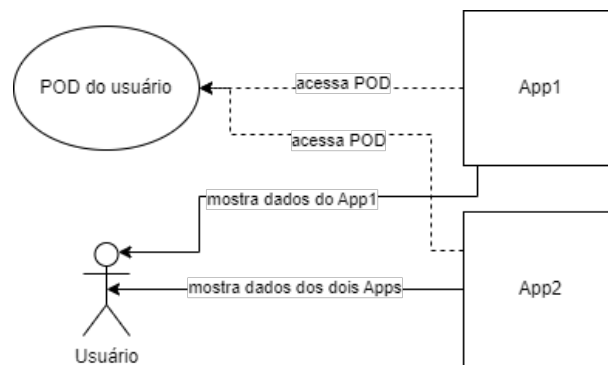


Figura 4.4: Diagrama Exemplificação de Interoperabilidade.

Anexo a este documento encontram-se *links* para os aplicativos em produção^{1 2} e também para os repositórios^{3 4} de seus respectivos projetos. Caso deseje executar os projetos localmente, após clonar os projetos dos repositórios e instalar as dependências, é necessário utilizar o comando `HTTPS=true npm start`, pois a autenticação com Solid exige um certificado web, mesmo rodando localmente.

Vale ressaltar que o objetivo deste trabalho não é fazer com que as aplicações façam parte da *smartUnB.ECOS* (pelo menos não neste primeiro momento), mas apenas servir como exemplo para que futuras aplicações possam usar os aprendizados daqui. Sendo assim, o design da UI é simples e não responsivo; logo, caso o leitor queira testar as aplicações, recomenda-se fazê-lo em um computador e não em celulares ou tablets.

¹<https://ruview.vercel.app/>

²<https://tutor-orcin.vercel.app/>

³<https://github.com/JomaSnow/Ruview>

⁴<https://github.com/JomaSnow/OwnOnOwn>

4.2 Implementação Solid

Para fazer os processos de autenticação do usuário com seu POD, autorização de privilégios da aplicação ao POD do usuário e as leituras e escritas em seus PODs, utilizamos as bibliotecas JavaScript da *Inrupt* (empresa de Berners-Lee): *@inrupt/solid-client*, *@inrupt/solid-client-authn-browser* e *@inrupt/vocab-common-rdf*.

Quanto aos testes, criamos PODs em 3 provedores diferentes para tentar simular 3 alunos diferentes utilizando as aplicações e interagindo entre eles. Utilizamos os seguintes provedores: *inrupt.net*, *solidcommunity.net* e *igrant.io*.

Sobre as funções de interação com os PODs, as duas aplicações são bem similares na implementação, uma vez que ambas utilizam JSONs para manusear seus dados. A exceção é para as funções que lidam com amigos e perfil do usuário, nas quais as manipulações são feitas usando os RDFs comuns providos pela biblioteca *@inrupt/vocab-common-rdf*, mas ainda assim, como essa funcionalidade é comum a ambas aplicações, então não têm tantas diferenças entre as aplicações em como elas interagem com os PODs dos usuários.

Vamos listar e descrever algumas funções mais importantes das aplicações a título de transparência a seguir, mas antes uma observação deve ser feita: não encontramos nenhuma informação sobre se a estrutura de um POD deve seguir algum padrão. Nos PODs dos provedores escolhidos, todos possuíam uma estrutura similar, com um diretório (Container) “profile” que contém um arquivo (DataSet) “card” no qual as informações (Properties) sobre o usuário são agrupadas (Things) e armazenadas. Outra informação contida neste arquivo é a de amigos. Desta forma, caso o usuário utilize um POD de um provedor que organiza os dados de forma diferente, ou caso ele crie seu próprio POD sem seguir essa mesma estrutura, a aplicação não funcionará corretamente. A mesma suposição foi feita sobre a presença de um diretório “public” no qual arquivos salvos dentro dele estão sempre disponíveis para leitura de outras aplicações sem necessidade de autorização do usuário (sendo necessária apenas para escrita).

Uma última observação é sobre a nomenclatura que a Inrupt utiliza para referenciar os dados dos PODs. Ela utiliza os conceitos de Things, DataSets e Containers, e os define da seguinte forma:

- **Thing:** Uma Thing pode ser entendida como uma referência a uma entidade do mundo real, similar ao conceito de objeto de Programação Orientada a Objeto. Os dados sobre aqueça Thing são chamados de propriedades. Por exemplo, em seu POD pode existir uma Thing “book1” com as propriedades “name” e “author”. Utilizam-se os vocabulários RDF para acessar as propriedades de cada Thing.

- **DataSet:** DataSets armazenam um conjunto de Things, e toda Thing precisa necessariamente fazer parte de um DataSet. DataSets podem ser entendidos como o arquivo em uma estrutura de diretórios.
- **Container:** Containers contêm DataSets e outros recursos, como outros Containers. Eles podem ser entendidos como sendo as pastas em uma estrutura de diretórios.

Autenticação

O processo de *login* e *logout* do ponto de vista de desenvolvimento é bem simples, e idêntico para ambas aplicações.

```

1 import {
2   getDefaultSession,
3   login,
4   logout,
5 } from "@inrupt/solid-client-authn-browser";
6
7 export async function solidLogin(
8   setWebIdSolidUpdateContextHook = () => {},
9   podUrl
10 ) {
11   if (!getDefaultSession().info.isLoggedIn) {
12     await login({
13       oidcIssuer: podUrl,
14       redirectUrl: window.location.href,
15       clientName: "Tutor",
16     });
17     setWebIdSolidUpdateContextHook(getDefaultSession().info.webId);
18   }
19 }
20
21 export async function solidLogout() {
22   await logout();
23   window.location.reload();
24 }

```

Listagem de Código 4.4: Funções de autenticação com Solid

A função *solidLogin* (Listagem 4.4, *linha 7*) usa como primeiro argumento uma função que persiste o webID do usuário em um state global através de *hooks* do React JS para que possa ser utilizado em todo projeto, e como segundo argumento a URL do provedor POD de escolha do usuário. Em sua execução, ela verifica se já existe algum usuário autenticado na *sessionStorage* da aplicação através de um método fornecido pela biblioteca da Inrupt. Caso não encontre informações de um usuário já autenticado, ela executa o método *login*

que também é fornecido pela biblioteca. Este método espera receber a URL do provedor e a URL da aplicação para redirecionar após concluir o processo de autenticação.

A função *solidLogout* (Listagem 4.4, *linha 21*) chama o método *logout* da biblioteca da Inrupt e em seguida atualiza a página. Este método apenas limpa as informações e tokens de usuário criados pelo *login* na *sessionStorage*, e a atualização da página é para atualizar todos os estados da aplicação.

Amigos

As aplicações buscam os amigos do usuário autenticado quando ele as acessa. Os amigos são salvos na propriedade “FOAF.knows” da Thing “me” do DataSet “card”. O webID do usuário automaticamente aponta para esse DataSet, então para acessar o DataSet basta informar o webID, e a Thing “me” é uma abreviação do webID do usuário dentro do DataSet (como se fosse uma variável chamada *me* recebendo como valor o webID do usuário).

```
1 //... inicio da funcao omitido para brevedade
2 const currentFriendsUrl = getUrlAll(thing, FOAF.knows);
3
4   for (const url of currentFriendsUrl) {
5     const friendDataSet = await getSolidDataset(url, { fetch: fetch })
6       ;
7
8     const friendThing = getThing(friendDataSet, url);
9
10    const nome = getLiteral(friendThing, FOAF.name).value;
11    const friendWebId = url;
12    const agenda = await getAgenda(url);
13    const compromissos = await getCompromissos(url, null);
14
15    const friendObj = { nome, friendWebId, agenda, compromissos };
16
17    friendsObjArr.push(friendObj);
18 //... resto da funcao omitido para brevedade
```

Listagem de Código 4.5: Trecho da função *getSolidFriends*

A função, agora com acesso ao Thing “me” do usuário, percorre todas os WebIDs armazenados em “FOAF.knows” para obter o webID de cada amigo. Em seguida, a função busca os arquivos desejados de cada aplicação e monta o objeto do amigo, retornando um *array* com todos os objetos de amigos. O objeto *fetch* (Listagem 4.5, *linha 5*) contém as informações de autorização que as funções da biblioteca Inrupt utilizam para validar se a aplicação está autorizada a acessar aquele POD.

Leitura de arquivos

A leitura de arquivos exige a URL do POD que contém o arquivo e o caminho até o arquivo. No *RUview*, por exemplo, para acessar o JSON de refeições curtidas, fica da seguinte forma:

```
1     const targetFileURL = podUrl + "public/ruview/likedMeals.json";
2
3     const likedMeals = await getFile(targetFileURL, { fetch: fetch });
4
5     const blob = await new Response(likedMeals).text();
6
7     const json = JSON.parse(blob);
8
9     return json;
```

Listagem de Código 4.6: Trecho de leitura de arquivos de PODs

A função utiliza o método *getFile* (Listagem 4.6, *linha 3*) para acessar o arquivo no caminho informado, utilizando o objeto *fetch* para verificar se o usuário tem autorização para fazê-lo. As demais linhas são apenas para converter o arquivo em JSON para que o JavaScript possa utilizá-lo (o método *getFile* retorna o arquivo em formato Blob). Essa lógica é utilizada em todas as funções que precisam acessar algum JSON no POD, como as de acesso à agenda, compromissos, refeições curtidas e refeições descurtidas.

Escrita e Remoção de arquivos

Tanto a escrita como a remoção de arquivos primeiro buscam os arquivos armazenados no POD, e em seguida utilizam as funções de manipulação de *arrays* do JavaScript para modificar os dados lidos.

```
1     const targetFileURL = podUrl + "public/ruview/likedMeals.json";
2
3     const fileArr = await getLikedMeals();
4
5     fileArr.push(meal);
6
7     await overwriteFile(
8       targetFileURL, // URL for the file.
9       JSON.stringify(fileArr), // File
10      { contentType: "application/json", fetch: fetch } // mimetype if
11      known, fetch from the authenticated session
12    );
```

Listagem de Código 4.7: Trecho de escrita de arquivos de PODs

Para a escrita, a função acrescenta o novo valor ao *array* utilizando o método *push*; para remoção, a função utiliza o método *filter* (Listagem 4.7, *linha 5*) para retornar um *array* sem o valor procurado. A função de atualização combina as duas: primeiro ela remove o valor procurado e depois insere o valor modificado. Para armazenar no POD, a função utiliza o método *overwriteFile* (Listagem 4.7, *linha 7*) da biblioteca *Inrupt* passando o novo *array*.

A função *getCompromissos*

Enquanto as funções do *RUview* são mais simples e diretas, devido à natureza mais complexa do *Tutor* as funções de arquivos precisaram ser mais complexas também. A função *getCompromissos*, embora tenha como objetivo final retornar um *array* com os compromissos do usuário, como as outras funções de leitura, ela acaba tendo que fazer um pouco mais, se tornando o “coração” do *Tutor*.

Não é possível escrever em um POD que não o seu. Sendo assim, uma dúvida surge: como vamos saber que um usuário recebeu uma proposta de reunião em determinado horário? Em um sistema centralizado poderíamos alterar os valores dos compromissos das pessoas envolvidas e todos teriam acesso às mesmas informações atualizadas imediatamente. Mas, como não estamos em um sistema centralizado, e a aplicação não armazena as informações sobre os compromissos de todos os usuários consigo, então informar essas interações entre usuários que precisam modificar os dados no POD que o usuário possui se torna um pouco trabalhoso.

A *getCompromissos* é responsável por esse trabalho, realizando algumas validações antes de retornar os compromissos do usuário. Ao ser chamada, ela já tem acesso aos compromissos de todos os amigos daquele usuário. Primeiro, ela resgata o JSON do POD do usuário. Em seguida ela apaga todas os compromissos que foram cancelados há mais de 2 semanas ou qualquer compromisso com mais de um mês. Isso é útil, pois diminui o custo da iteração nos *arrays* de compromissos. Depois ela compara os compromissos de cada amigo e verifica:

- se tem algum compromisso novo com status 0 (pendente) nos compromissos do amigo, então ela cria um novo compromisso no mesmo dia com status 4 (aguardando sua resposta) no POD do usuário
- se tem algum compromisso em ambas agendas na qual o status seja 2 (cancelado por você) no compromisso amigo, então modifica o compromisso para status 3 (cancelado pelo amigo) no POD do usuário

- se tem algum compromisso em ambas agendas na qual o status seja 1 (confirmado) no compromisso do amigo, então modifica o compromisso para status 1 (confirmado) também no POD do usuário

Essas validações garantem que o usuário sempre terá a resposta mais atualizada e permite exibir notificações na página inicial e popular corretamente a agenda com os compromissos pendentes e confirmados.

4.3 Testes alfa com as aplicações desenvolvidas

Iremos, nesta seção, expandir nas funcionalidades mais interessantes das aplicações desenvolvidas, aprofundando nos comportamentos descritos pela Figura 4.1, Figura 4.2 e Figura 4.3. Vamos descrever alguns casos de uso das aplicações e exibir recortes das *screenshots* que mostram as aplicações em funcionamento. Utilizamos os cenários descritos aqui para testar o funcionamento das aplicações.

4.3.1 RUview

Atualização de cardápios por funcionário da UnB

Um funcionário responsável por criar, modificar e manter as refeições e cardápios dos restaurantes comuns e executivo pode acessar a área de administrador da aplicação. Essa área pode ser vista na Figura 4.5. Nela, o funcionário responsável pode adicionar uma refeição nova, informando o nome da refeição, em qual momento ela será servida (café, almoço ou jantar) e em qual restaurante (refeitório ou executivo), além de poder modificar os pratos de cada dia no cardápio. Os pratos e cardápios são armazenados no servidor do Firebase, e toda alteração feita é refletida diretamente para os alunos fora da área de administrador da aplicação, uma vez que essa parte das refeições e cardápios se trata de armazenamento de dados de forma centralizada.

Aspecto social do RUview

Na tela inicial, exibida na Figura 4.6, os pratos do dia são exibidos de maneira organizada; café da manhã, almoço e jantar de cada restaurante: refeitório ou executivo. A aplicação busca esses dados do servidor do Firebase no momento que ela é iniciada.

Esse aluno pode optar por permitir à aplicação o acesso a seu POD para ver quais amigos demonstraram interesse em algum prato daquele dia e ajudá-lo a decidir se irá comer no *campus* ou não. Essas avaliações de cada amigo vêm dos PODs de cada um. Este aluno pode, também, avaliar algum prato que gosta com um “gostei”, o que irá

Cardápio
REFEITÓRIO ▾

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo
Café da manhã	Salsicha ao ▾	Carne desfi ▾	Queijo mina ▾	Ovos mexidi ▾	Bebida lácte ▾	Queijo muçz ▾	Franco desfi ▾
Almoço	Cubos de cz ▾	Franco assa ▾	Carne de so ▾	Filé de franc ▾	Lasanha à b ▾	Filé de franc ▾	Almôndegas ▾
Janta	Filé de franc ▾	Jardineira de ▾	Cubos suínc ▾	Isca de carn ▾	Franco assa ▾	Assado de c ▾	Filé de peix ▾

Atualizado por último em: Tue Jun 13 2023 21:22:44 GMT-0300 (Brasília Standard Time) Atualizar Cardápio

Cadastrar Nova Refeição

Nome

Selecione... ▾


Selecione... ▾

- Contém leite
- Contém cogumelo
- Contém mel
- Contém oleaginosas
- Contém ovos
- Contém pimenta
- Contém soja
- Contém suíno
- Contém trigo

Choose File | No file chosen


Adicionar

Refeições Cadastradas



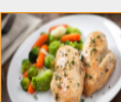
Frango desfiado

Contém:
Nenhum atributo



Salsicha ao molho

Contém:
Suíno



Filé de frango ao molho oriental

Contém:
Soja

Figura 4.5: Tela de administrador do RUview.

escrever em seu POD que ele gosta daquela refeição, bem como o dia da semana e qual horário ela é servida. Esta última parte tornará possível a interoperabilidade entre as aplicações. É possível ainda que o aluno volte atrás, retirando aquele prato de seu POD, e em seguida mude sua avaliação original.

4.3.2 Tutor

Edição da agenda

O aluno deve permitir o acesso do Tutor para poder visualizar e atualizar sua agenda. Ele pode definir horários livres e indisponíveis. A aplicação define o *status* das demais células com base nos compromissos de cada dia. Tanto a agenda quanto os compromissos do aluno são armazenados em seu POD. A Figura 4.7 mostra como é a edição da agenda no Tutor. Os dias e horário definidos como livres são usados para validar as demais funcionalidades da aplicação, como agendar nova reunião e exibição da mensagem de sugestão de reunião

Pratos do dia

Terça-feira. (Atualizado em 13/06/2023, 21:22:44)

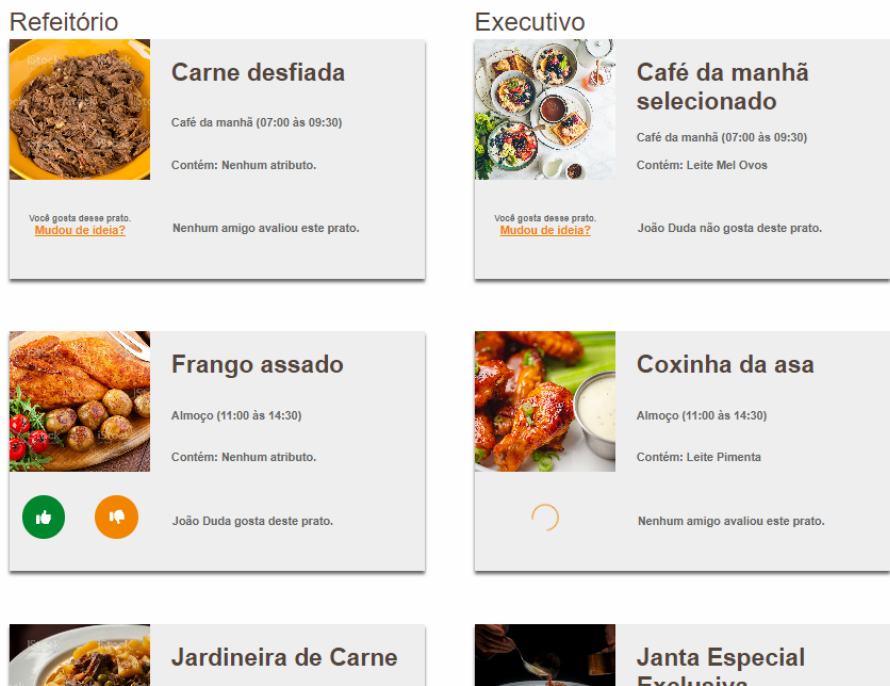


Figura 4.6: Tela com os pratos do dia do RUview.

com base no uso do RUview, de modo que a edição da agenda do aluno deve ser a primeira etapa a ser realizada.

Reunião com amigo

A funcionalidade principal da aplicação, solicitar reunião com amigo, trabalha a partir da busca da agenda no POD do amigo, porém modificada para impossibilitar a interação de células inativas e trocando células “pendentes” por células “livres”. A Figura 4.8 mostra o processo de solicitar reunião com o amigo quando interage com uma célula “livre”.

O aluno pode escolher de qual amigo ele quer ver a agenda e Tutor vai buscar e representar os horários deste amigo corretamente, mas na realidade a aplicação busca os dados do POD de cada amigo assim que inicializa, incluindo a agenda de cada um. Nesta etapa de busca no POD do amigo, a aplicação também pega os pratos curtidos daquele amigo no RUview para informar caso ambos alunos tenham disponibilidade naquele dia e horário da refeição que demonstraram interesse no RUview. Ao enviar a solicitação, a

Agenda

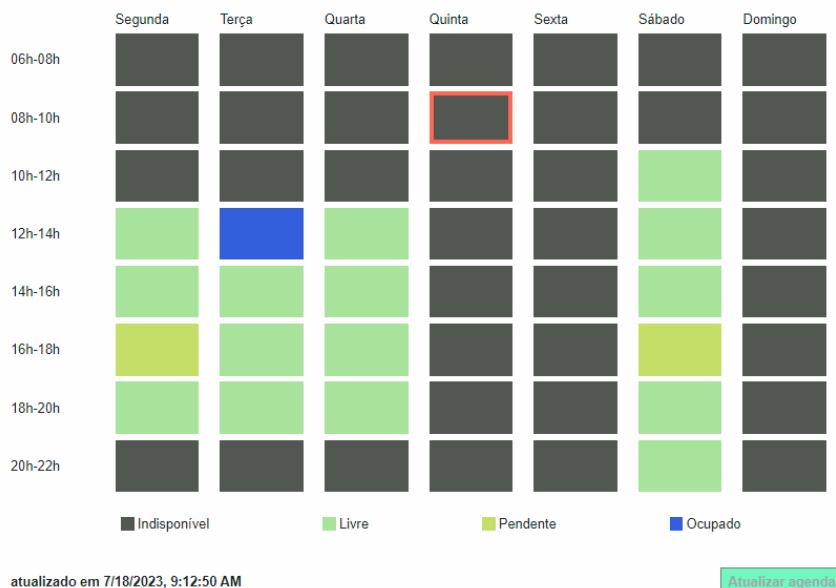


Figura 4.7: Tela de edição da agenda do Tutor.

aplicação escreve um novo compromisso no POD do aluno, e o *getCompromissos* do aluno e do amigo trabalham para garantir dados atualizados entre os PODs, como vimos na seção 4.2.

Reuniões agendadas

O aluno pode visualizar todas as atualizações em volta do envio e recebimento de solicitações de reuniões, sendo listas das mais recentes para as mais antigas, como mostra a Figura 4.9. Na realidade, a aplicação está usando a *getCompromissos* para buscar os compromissos do POD do aluno e sincronizá-los com os compromissos dos PODs dos amigos. Cada compromisso (reunião) tem ações para confirmar ou recusar tal compromisso e atualizar no POD do aluno.

Com as aplicações finalizadas, e com um entendimento maior do processo de desenvolvimento e das funcionalidades delas, podemos avaliar os resultados obtidos.

Agendar nova reunião

Selecione um parceiro de estudo para ver sua agenda da semana.

Agenda de João Duda Vocês dois gostam do almoço do RU de hoje! Por que não reunir no campus hoje, às 12h-14h?

Terça-Feira, 12h-14h ✕

Este horário está disponível. Você pode enviar uma proposta para marcar uma reunião com este amigo.

[Solicitar reunião](#)

Horário	Seg	Ter	Qua	Qui	Sex	Sab	Domingo
06h-08h	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível
08h-10h	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível
10h-12h	Livre	Livre	Livre	Livre	Livre	Livre	Indisponível
12h-14h	Livre	Livre	Livre	Livre	Livre	Livre	Indisponível
14h-16h	Livre	Livre	Livre	Livre	Livre	Livre	Indisponível
16h-18h	Livre	Livre	Livre	Livre	Livre	Livre	Indisponível
18h-20h	Livre	Livre	Livre	Livre	Livre	Livre	Indisponível
20h-22h	Indisponível	Indisponível	Indisponível	Indisponível	Indisponível	Livre	Indisponível

■ Indisponível ■ Livre ■ Ocupado

Figura 4.8: Tela solicitação de reunião na agenda do amigo do Tutor.

Reuniões agendadas

- Você enviou uma proposta de reunião Segunda-Feira, 16h-18h para João Duda. ✗ Cancelar
- Reunião confirmada Terça-Feira, 12h-14h com João Duda. ✗ Cancelar
- João Duda quer reunir Sábado, 16h-18h. ✓ Confirmar ✗ Cancelar

Agendar nova reunião

Selecione um parceiro de estudo para ver sua agenda da semana.

Agenda de João Duda

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado	Domingo
06h-08h	█	█	█	█	█	█	█
08h-10h	█	█	█	█	█	█	█
10h-12h	█	█	█	█	█	█	█
12h-14h	█	█	█	█	█	█	█
14h-16h	█	█	█	█	█	█	█

Figura 4.9: Tela com as reuniões agendadas do Tutor.

Capítulo 5

Resultados

Para este trabalho, nos propusemos a primeiro avaliar a adoção da tecnologia Solid e em seguida avaliar como aplicações Solid poderiam agregar a um contexto educacional e a experiência de desenvolvimento com esta tecnologia. Neste capítulo iremos exibir os resultados desses objetivos.

5.1 Adoção

Para o primeiro objetivo, após realizar a pesquisa e mapeamento de algumas aplicações Solid-compatíveis, e interpretando as informações da Tabela 3.1, pudemos chegar em alguns resultados. A primeira observação notável diz respeito à quantidade de aplicações com viés comercial. A Figura 5.1 mostra 11 aplicações que julgaram que o Solid está avançado o suficiente para ser utilizada profissionalmente, ou que acreditam que a tecnologia trará resultados no futuro. É claro que, quando comparado à quantidade de aplicações comerciais que existem, esse número é um tanto pequeno. Porém, ele mostra que existem empresas dispostas a dar uma chance ao Solid.

Uma outra análise interessante é a dos países que estão produzindo as soluções em Solid. A Figura 5.2 mostra a distribuição dos países que desenvolveram as aplicações (Espanha ajustada para incluir apenas uma entrada dos projetos da Universidade de Oviedo). Percebemos primeiramente a falta de países da América Latina, o que mostra que a *smartUnB.ECOS* pode ser pioneira em sua adoção do Solid. É interessante também a concentração europeia neste gráfico, com apenas uma aplicação dos Estados Unidos não fazendo parte desta concentração. Isso pode ser influenciado pelos governos e grandes empresas europeias estarem estimulando a adoção do Solid, como por exemplo o caso do governo de Flandres e dos PODs da BBC.

Um outro recorte que podemos fazer diz respeito a quais os tipos de serviços que as aplicações prestam. A Figura 5.3 mostra esse recorte. Podemos observar uma grande

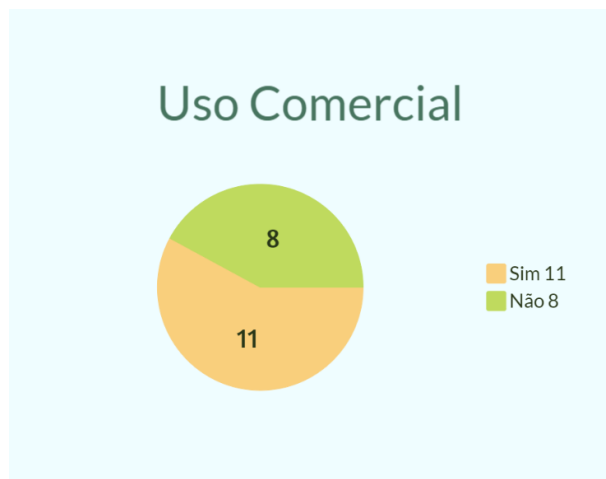


Figura 5.1: Gráfico Uso Comercial.

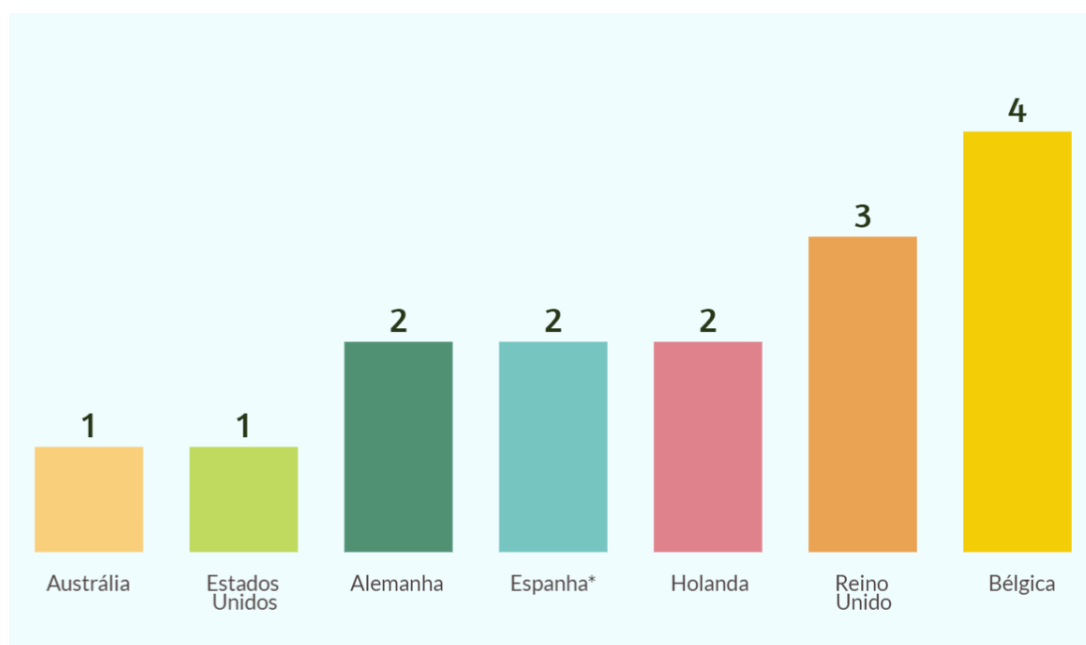


Figura 5.2: Gráfico Distribuição por País.

quantidade de aplicações de demonstração da tecnologia, o que faz sentido para uma tecnologia que ainda está em desenvolvimento como é o caso do Solid. Desconsiderando essas, podemos observar uma certa tendência a aplicações na área da saúde e aplicações de fornecimento de PODs. Certamente a notoriedade do projeto do NHS britânico contribuiu para que mais empresas utilizassem Solid para aplicações de saúde; e quanto ao fornecimento de PODs, uma mentalidade “na corrida do ouro, enriquece quem vende pás” faz sua parte também.

Em uma última análise, podemos observar na Figura 5.4 quantas aplicações foram



Figura 5.3: Gráfico Tipos de Serviços.

publicadas em cada ano. O ano de 2023 não tem mais aplicações uma vez que a seleção destas aplicações para este trabalho concluiu antes da metade do ano progredir. Sendo assim, é possível notar um modesto crescimento em aplicações que estão utilizando o Solid, inclusive com o projeto da BBC que ainda não tem uma data de lançamento. Isso sugere que, embora devagar, o Solid vem alcançando certa adoção por parte da comunidade.



Figura 5.4: Gráfico Projetos por Anos.

5.2 Desenvolvimento

Quanto ao segundo objetivo do trabalho, fomos capazes de desenvolver duas aplicações Solid-compatíveis voltadas ao público alvo da *smartUnB.ECOS* (ou seja, alunos e professores do CIC). Utilizamos bibliotecas em JavaScript fornecidas para desenvolvedores pela Inrupt, empresa fundada por Berners-Lee, criador do Solid; isso mostra que a tecnologia quer ser adotada pela comunidade de desenvolvimento e vai fazer o possível para facilitar essa adoção.

No desenvolvimento, elaboramos algumas funções para fazer as conexões com os PODs dos usuários e exibir seus dados na aplicação. No entanto, encontramos algumas dificuldades no processo, e nossas aplicações têm algumas limitações como consequência. Vamos compartilhar alguns dos aprendizados a seguir.

Originalmente, o plano para atingir o segundo objetivo deste trabalho era de desenvolver aplicações mobile que fizessem uso do Solid, pois isso se adequaria melhor ao cotidiano dos estudantes, que nem sempre têm um computador a disposição, mas nunca saem sem seus celulares. Entretanto, não conseguimos fazer a biblioteca da Inrupt fazer o processo de autenticação nos dispositivos móveis, uma vez que ela primeiro redireciona para o provedor desejado, e depois retorna e salva os tokens na *sessionStorage* da aplicação. Para utilizar essa biblioteca em mobile, teríamos que abrir o navegador do celular de dentro da aplicação e também modificar a própria função de *login* da biblioteca da Inrupt para não tentar acessar o *sessionStorage* (uma vez que diferentes aparelhos armazenam de formas diferentes), mas sim retornar todos os dados para o desenvolvedor salvá-los como achar necessário. Não encontramos outras bibliotecas que realizassem esse processo.

Embora a documentação de uso das bibliotecas da Inrupt sejam suficientes, a comunidade de desenvolvimento Solid deixa a desejar, no sentido de fazer conteúdo educacional em volta do desenvolvimento com Solid, como encontra-se de outras tecnologias, como vídeos e postagens em fóruns. Os projetos da Universidade de Oviedo foram um achado valiosíssimo, pois além de *open-source*, os alunos documentaram todo o processo de desenvolvimento de todas as aplicações, sendo de muita ajuda quando encontramos alguma dificuldade de desenvolvimento e notamos que eles também encontraram e já tinham a solução. Mais recursos de apoio voltados aos desenvolvedores faz falta quando trabalhando com uma tecnologia nova.

Uma outra dificuldade encontrado foi com a própria biblioteca utilizada. As bibliotecas da Inrupt exigem o uso do NPM a versão exata 16.0.0, que em conjunto com as demais bibliotecas utilizadas em desenvolvimento, apresenta uma vulnerabilidade grave, segundo os logs feitos no console sempre que se executa a aplicação localmente. Não é claro porque eles não mantêm as bibliotecas atualizadas para as versões mais recentes do NPM, mas

como essas bibliotecas são essenciais para o desenvolvimento, fomos forçados a fazer um *downgrade* para essa versão mais antiga.

Um obstáculo já mencionado no Capítulo 4 é a impossibilidade de escrever em PODs que não os do usuário autenticado, ainda que em Containers públicos. Isso faz com que seja trabalhoso manter dados atualizados entre dois ou mais usuários (pois, precisam realizar todas as validações por conta ao iniciar a aplicação; vide *getCompromissos*). Em sistemas centralizados, isso não é nenhum problema, uma vez que o servidor sempre tem acesso aos dados atualizados de todos os usuários.

Isso nos leva ao próximo obstáculo, o que julgamos ser o mais sério: o acesso aos PODs podem falhar. Em aplicações centralizadas, todas as transações são atômicas, ou seja, ou uma busca/escrita de dados é feito por completo, ou nada é feito. Mas no caso de aplicações descentralizadas, como os dados estão espalhados ao longo de vários PODs, o que acontece se o acesso a deles falha? A aplicação não pode retornar somente os dados que conseguiu acessar, pois isso resultaria em exibir dados desatualizados. A aplicação pode cancelar toda a transação, similar ao que é feito em aplicações centralizadas, mas isso geraria falhas cada vez mais constantes a medida que o número de diferentes PODs necessários for aumentando. A aplicação também pode armazenar os dados em seus servidores, mas isso efetivamente vai contra os princípios do Solid, e se for ter que chegar a isto é melhor utilizar soluções centralizadas direto.

Quanto a limitações das aplicações desenvolvidas, existem algumas. A primeira, como já foi comentado no capítulo anterior, é que nossas aplicações, assim como várias outras aplicações de código aberto estudadas, assumem uma determinada estrutura na organização dos PODs dos usuários, e não encontramos a existência de um padrão a ser seguido ao criar PODs. Outra limitação sobre a forma que implementamos nossas funções é que elas são executadas linearmente, um comando após o outro. Isso faz com que funções que precisam acessar o POD de cada amigo do usuário fique cada vez mais demorada à medida que o número de amigos aumenta. Uma última limitação é assumir que o usuário não fará alterações aos dados das aplicações fora delas (diretamente em seu POD). Por exemplo, um amigo pode recusar a reunião com o usuário, resultando em um status 3 (cancelado pelo amigo), como visto na Tabela 4.2, mas o usuário pode em seguida alterar o status em seu POD para 1 (confirmado), e a aplicação quando o amigo acessar vai observar esse status e modificar o POD do amigo para confirmar a reunião para ele também.

Dito isso, apesar das dificuldades e limitações, fomos capazes de desenvolver as aplicações para um contexto social-educacional e, apesar da experiência de desenvolvimento poder ser melhor, a tecnologia já consegue ser utilizada em aplicações menores como as desenvolvidas aqui. Sendo assim, acreditamos que a tecnologia agregaria a um contexto social-educacional como o *smartUnB.ECOS*, e que os aprendizados obtidos deste trabalho

auxiliarão nessa adoção.

Capítulo 6

Conclusão

Conseguimos mostrar neste trabalho que o Solid vem aumentando em sua adoção, especialmente ao longo de 2022 e em área da saúde e de provedores de PODs, com grandes empresas como a BBC com previsão de lançar projetos no futuro. No entanto, levando em conta que a tecnologia foi divulgada em 2016 e até agora existem tão poucas implementações, indica que ela ainda é bastante nicho, sendo necessária uma maior divulgação de seus pontos positivos (ou de repente dos pontos negativos da Web centralizada de hoje para convencer a mudança de paradigma) se for para se tornar a tecnologia disruptiva que ela almeja ser.

Do ponto de vista do público maior, existe pouco incentivo para a adoção do Solid. Enquanto as grandes empresas forem centralizadas e controlarem todos os dados que os usuários utilizam cotidianamente, especialmente quando a maior parte das pessoas não vão muito longe das empresas grandes, os PODs seriam mais uma fonte de dados que ele mesmo precisaria cuidar (na prática mais um aplicativo que o usuário seria esperado a usar) sem ter como tirar todo o proveito da tecnologia devido a falta de aplicações Solid-compatíveis e a falta da adoção do Solid por parte das grandes empresas.

Quanto aos desenvolvedores, a tecnologia tem seus pontos fracos, por exemplo, atualmente é impossível utilizar Solid offline, e por ainda ser uma tecnologia nova, ainda não existem tantos recursos educacionais por parte da comunidade, como vídeos ou fóruns auxiliando o desenvolvimento. Apesar disso, hoje já é possível desenvolver aplicações Solid-compatíveis para um contexto social-educacional, e fomos capazes de mostrar isso neste trabalho. No entanto, desenvolvedores ainda precisam de mais estímulos para criar as aplicações que trarão os usuários para a descentralização, como mais bibliotecas e ferramentas que facilitem o trabalho de desenvolvimento e conteúdos em volta do desenvolvimento com Solid. A adoção do Solid no *smartUnB.ECOS* pode prover esse estímulo.

Referências

- [1] Nóbrega, Germana, Gabriel Silva e Thiago Silva: *Um projeto estruturante para orientações de tcc em cursos de computação: que oportunidades para ihc?* Em *Anais do XIII Workshop sobre Educação em IHC*, páginas 19–24, Porto Alegre, RS, Brasil, 2022. SBC. <https://sol.sbc.org.br/index.php/weihc/article/view/22854>. viii, 2, 9, 10
- [2] Louie Andre: *53 important statistics about how much data is created every day*, 2023. <https://financesonline.com/how-much-data-is-created-every-day/>, Acessado por último em 08/07/2023. 1
- [3] Eliza Crawford: *Website tracking: Why and how do websites track you?*, 2020. <https://www.cookiepro.com/blog/website-tracking/>, Acessado por último em 08/07/2023. 1, 5
- [4] Bottles, Kent, Edmon Begoli e Brian Worley: *Understanding the pros and cons of big data analytics*. *Physician executive*, 40(4):6–12, 2014. 1
- [5] Li, Yinghao, Qiumei Pu, Shuheng Li, Hong Zhang, Xiaofeng Wang, Haodong Yao e Lina Zhao: *Machine learning methods for research highlight prediction in biomedical effects of nanomaterial application*. *Pattern Recognition Letters*, 117:111–118, 2019. 1
- [6] Caitlin Jones: *50 phishing stats you should know in 2023*, 2023. <https://expertinsights.com/insights/50-phishing-stats-you-should-know/>, Acessado por último em 08/07/2023. 1
- [7] The Economist: *The world’s most valuable resource is no longer oil, but data, The data economy demands a new approach to antitrust rules*, 2017. <https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data>, Acessado por último em 08/07/2023. 1
- [8] Young, Elise: *Educational privacy in the online classroom ferpa moocs and the big data conundrum*. *Harvard Journal of Law Technology* Volume 28, Number 2, páginas 549—592, 2015. 1, 5, 6
- [9] Verborgh, Ruben: *Re-decentralizing the Web, for good this time*. Em Seneviratne, Oshani e James Hendler (editores): *Linking the World’s Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*. ACM, 2022. <https://ruben.verborgh.org/articles/redecentralizing-the-web/>. 2, 6, 14

- [10] Kola, Lola, Brandon A Kohrt, Charlotte Hanlon, John A Naslund, Siham Sikander, Madhumitha Balaji, Corina Benjet, Eliza Yee Lai Cheung, Julian Eaton, Pattie Gonsalves *et al.*: *Covid-19 mental health impact and responses in low-income and middle-income countries: reimagining global mental health*. The Lancet Psychiatry, 8(6):535–550, 2021. 2
- [11] Yeyati, Eduardo Levy, Federico Filippini *et al.*: *Social and economic impact of covid-19*. Brookings Institution, 2021. 2
- [12] Cida de Oliveira: *Os sete erros de bolsonaro que permitiram 75/100 das 690 mil mortes por covid no brasil*, 2022. <https://www.redebrasilatual.com.br/saude-e-ciencia/sete-erros-bolsonaro-covid-brasil/>, Acessado por último em 08/07/2023. 2
- [13] Tribunal Superior Eleitoral: *Por maioria de votos, tse declara bolsonaro inelegível por 8 anos*, 2023. <https://www.tse.jus.br/comunicacao/noticias/2023/Junho/por-maioria-de-votos-tse-declara-bolsonaro-inelegivel-por-8-anos>, Acessado por último em 08/07/2023. 2
- [14] Sérgio de Andrade Nishioka: *O uso de máscaras continuará frequente depois de terminada a pandemia?*, 2021. <https://www.unasus.gov.br/especial/covid19/markdown/441>, Acessado por último em 08/07/2023. 2
- [15] Pokhrel, Sumitra e Roshan Chhetri: *A literature review on impact of covid-19 pandemic on teaching and learning*. Higher education for the future, 8(1):133–141, 2021. 2
- [16] Maatuk, Abdelsalam M, Ebitisam K Elberkawi, Shadi Aljawarneh, Hasan Rashaideh e Hadeel Alharbi: *The covid-19 pandemic and e-learning: challenges and opportunities from the perspective of students and instructors*. Journal of computing in higher education, 34(1):21–38, 2022. 2
- [17] Ana Luiza Figueiredo: *Primeira multa por violação da lgpd é aplicada após 3 anos*, 2023. <https://olhardigital.com.br/2023/07/06/pro-primeira-multa-por-violacao-da-lgpd-e-aplicada-apos-3-anos/>, Acessado por último em 08/07/2023. 5
- [18] Wan-Shiou Yang, Jia-Ben Dia, Hung Chi Cheng Hsing Tzu Lin: *Mining social networks for targeted advertising*. Proceedings of the 39th Hawaii International Conference on System Sciences, 2006. 5
- [19] Lekha R. Nair, Sujala D. Shetty, Siddhant Deepak Shetty: *Streaming big data analysis for real-time sentiment based targeted advertising*. International Journal of Electrical and Computer Engineering (IJECE) Vol. 7, No. 1., páginas 402—407, 2017. 5
- [20] Katie O’Donnell, Henriette Cramer: *People’s perceptions of personalized ads*. International World Wide Web Conference Committee (IW3C2), páginas 1293–1298, 2015. 5

- [21] Charles Duhigg: *How companies learn your secrets*, 2012. <https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html>, Acessado por último em 08/07/2023. 5
- [22] Charoula Angeli, Sarah Katherine Howard, Jun Ma Jie Yang Paul A. Kirschner: *Data mining in educational technology classroom research: Can it make a contribution?* University of Wollongong, 2017. 5
- [23] Essam Mansour, Andrei Vlad Sambra, Sandro Hawke Maged Zereba Sarven Capadisli Abdurrahman Ghanem Ashraf Aboulnaga Tim Berners Lee: *A demonstration of the solid platform for social web applications*. WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada., páginas 223—226, 2016. 6, 8
- [24] Steve Lohr: *He created the web. now he's out to remake the digital world.*, 2021. <https://www.nytimes.com/2021/01/10/technology/tim-berners-lee-privacy-internet.html>, Acessado por último em 08/07/2023. 7
- [25] World Wide Web Foundation: *Three challenges for the web, according to its inventor*, 2017. <https://webfoundation.org/2017/03/web-turns-28-letter/>, Acessado por último em 08/07/2023. 7
- [26] Mainini, Pascal e Annett Laube-Rosenpflanzler: *Access control in linked data using webid*. arXiv preprint arXiv:1611.03019, 2016. 8
- [27] Bizer, Christian, Tom Heath e Tim Berners-Lee: *Linked data: The story so far*. Em *Semantic services, interoperability and web applications: emerging concepts*, páginas 205–227. IGI global, 2011. 8
- [28] Decker, Stefan, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann e Ian Horrocks: *The semantic web: The roles of xml and rdf*. IEEE Internet computing, 4(5):63–73, 2000. 8
- [29] Berners-Lee, Tim, James Hendler e Ora Lassila: *The semantic web*. Scientific american, 284(5):34–43, 2001. 8
- [30] Harper, Corey A e Barbara B Tillett: *Library of congress controlled vocabularies and their application to the semantic web*. Cataloging & classification quarterly, 43(3-4):47–68, 2007. 8
- [31] World Wide Web Consortium: *vcard ontology - for describing people and organizations*, 2014. <https://www.w3.org/TR/vcard-rdf/>, Acessado por último em 10/07/2023. 8
- [32] Brickley, Dan e Libby Miller: *Foaf vocabulary specification 0.91*, 2007. 8
- [33] Stack Overflow: *2023 developer survey*, 2023. <https://survey.stackoverflow.co/2023/#technology-most-popular-technologies>, Acessado por último em 08/07/2023. 11

Apêndice A

Links das aplicações desenvolvidas

- *RUview produção* - <https://ruview.vercel.app/>
- *RUview repositório* - <https://github.com/JomaSnow/Ruview>
- *Tutor produção* - <https://tutor-orcin.vercel.app/>
- *Tutor repositório* - <https://github.com/JomaSnow/OwnOnOwn>