A Search for a Better Search

Michał Poręba

Department for International Trade
Swansea University

1 Where has it all started?

On the 14th of October 2010, Martha Lane Fox, UK Digital Champion, wrote to Francis Maude, then Minister for the Cabinet Office, a letter [1]. In that letter, after reviewing *Directgov*[1], the precursor to what is now *gov.uk*[2], she called for revolution, not evolution in the way the government services are provided to people living in the UK.

She wrote: "There has been a reinvention of the Internet and the behaviour of users in the last few years. Digital services are now more agile, open and cheaper. To take advantage of these changes, government needs to move to a 'service culture', putting the needs of citizens ahead of those of departments. This increase in focus on end users should include opening up government transactions so they can be easily delivered by commercial organisations and charities, and putting information wherever people are on the web by syndicating content" [2].

The report went on to show how few people get to the services or data they need from the home page. It concluded that efforts should be made so that "citizens can find what they want wherever they are on the web" [2].

It led to the creation of Government Digital Service (GDS), an organisation within the Cabinet Office, with a mission to revolutionise our interactions with the government [3].

1.1 Digital Government

The digitalisation of citizen services is not just a British phenomenon. It is known under many names: Digital Government, e-Government, Government 2.0. They all mean more or less the same.

In 2003 the Organisation for Economic Co-operation and Development (OECD) published the "e-Government Imperative" report, which already[3] acknowledged that the fastly developing *web* was changing expectations not only of consumers towards business offerings but also those of citizens towards their governments.

Digital Government is a concept capturing this relatively new, digital way of citizens interacting with their governments. It became a very enticing option for many governments, especially after the financial crisis of 2008, but despite a lot of interest and early surge, the adoption stalled [4]. After researching the problem, the OECD proposed changing the focus from the digitisation of existing processes to focusing on the users, the citizens, and their needs.

The GDS and the UK's Digital Strategy [5] fit well in that idea of user-centric Digital Government, and they are working. The Digital Government Index 2019 published by the OECD in 2020 showed the United Kingdom is one of the better-performing countries across all six domains and as the leader in two of them! [6]

1.2 The wider perspective

But, that was just the beginning. Over the last decade, the user-centric approach championed by the GDS achieved a lot. The increase in the number of services available online is breathtaking. At the same time, it resulted in many disjoint services and data silos. The distinctive, uniform look and feel of *gov.uk* gives an impression of a single service. Nonetheless, behind the scenes, thousands of independent services are implemented in various technologies, hosted by individual departments, using variety of platforms. While more services and data can be accessed online, there is still much more to do to provide a better, joined-up user experience and to complete the revolution called for in 2010.

Realising that, the Cabinet Office created the Central Digital and Data Office (CDDO) in April 2021 with a mission to deliver the digital transformation at scale, and to build, support and improve digital and data products that can work across government [7].

The emphasis on cross-government functionality of the services is essential to reduce cost and increase value both to individual citizens and the government. However, while the problem of disjoint data silos is obvious, the solution is not necessarily so. For years the go-to solution for data was centralisation. Data warehouses, data lakes, data lakehouses [8]. However, as the fourth industrial revolution progresses, as we generate more and more data, we become more sensitive to data security and ethical concerns around it [9]. Rightfully so. But, it means that data centralisation looks less and less like the answer we are looking for [10]. We need alternatives.

In the service layer, the monolith architecture gave way to microservice design. While this approach helped to achieve the progress we have witnessed in the last ten years it doesn't address the challenges of distributed, federated systems. If anything, it makes things harder.

1. at the time government website providing transactional services like student loans, jobseekers allowance, or car tax
2. currently the digital door of UK government
3. 4 years before iPhone, the same year G3 was invented!

2 The federated idea

To explore the challenges of working with data across the Civil Service, I proposed a *find me button*. The idea is trivial. There is already a search service on *gov.uk*. It allows the public to search for text in all the officially published documents. I wanted to add a *find me now* action button as in Figure 1 which would allow, with a single click, to discover all of the information the government holds on us[4].

I had two main reasons to start with this idea despite how unlikely it may sound. Firstly, under the UK General Data Protection Regulation (UK GDPR) from the Data Protection Act 2018 [11] it is our right to request the data about us and be able to amend or rectify it if necessary [12]. It is our right, and it is already possible. It just requires many letters and even more patience. It is expensive and time-consuming for both the individual and the government. Surely we can do better in the 2020s!

Secondly, this simple, easy-to-visualise idea exposes many problems and perfectly illustrates how far we are from fully realising our ambition of the Digital Government.

Now, after ten months of looking into the problem, I am convinced that it also shows what could be possible if only we looked at the problem from a slightly different technical perspective.

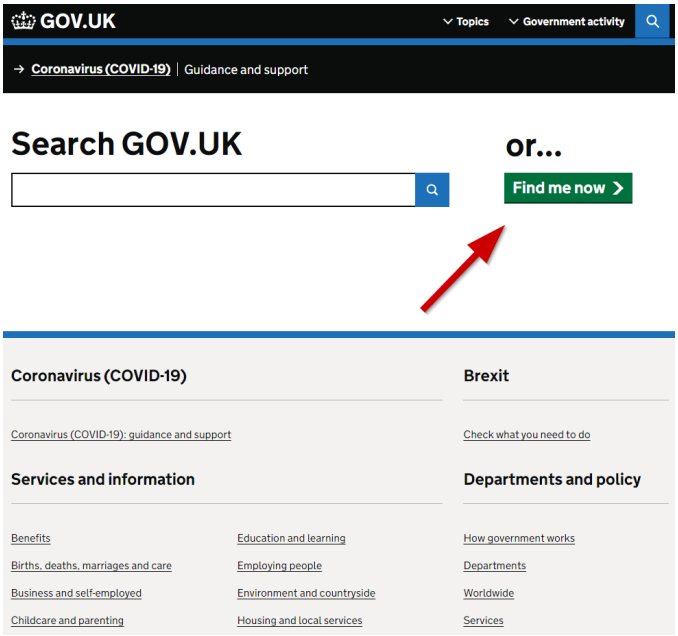

Figure 1. The *find me button* visualisation

2.1 The challenges

Before we look at the solution, let us consider the problem domain and why it is so difficult.

When I started the research last autumn, the Civil Service, the organisation that holds the data about us, consisted of 449 departments, agencies and public bodies [13]. Right now, the number is 456 and still, it does not include local authorities or devolved administrations.

4. of course with the necessary permissions so that each of us can look up information only about ourselves and nobody's else

There is no one organisation holding our data. Instead, it is a federated mesh of largely independent organisations with independent priorities, practices, procedures, and purses.

Standardisation - one of the possible solutions - of anything at this scale is difficult, if not impossible. It is especially true in an environment where the politics of the day influence the development priorities. Nevertheless, even without it in a decentralised organisation, who should be setting the standards? Who should be adopting them? What to do with existing technology? What is the incentive to change what we already have?

Centralisation - another possibility - could clear the technical hurdles. After all, we can do big data. However, the legal aspect proves very difficult. Who is responsible for the safekeeping and safeguarding of the data in that central store? Is it the department that collected it? Is it the one that stores it? Or is it the one that is using it?

Legislation - a unique opportunity to the Civil Service - is a way to alter the law to mandate something is possible in a specific way or to make the lack of a working solution illegal. This approach could lead the quickest to a change. However, the current trajectory of the legal process is strongly influenced by our growing understanding of data ethics, so data centralisation is likely to become more difficult rather than easier in the future.

Our ever-increasing concern for data ethics makes the problem more challenging in all of the above approaches. The stakes are high. Getting the balance wrong can lead to insufficient innovation or social rejection of otherwise working solutions [9].

The challenges are widely accepted. The CDDO has been created to address them. In the policy paper *Transforming for a digital future: 2022 to 2025 roadmap for digital and data* published in June 2021 we can read about challenges ahead: "The UK government still lags behind other sectors. Our services are often slow, difficult to use and expensive to deliver. [...] Data quality is inconsistent and frequently poor and effective data sharing between departments is limited. [...] We have significant challenges to overcome. We need to address years of uneven progress and siloed development in individual departments [...]." [14]

2.2 The established solutions

We can get a sense of how real, how big the problem is by the abundance of attempts to solve it. A quick online search reveals many big projects already in progress.

The Government Data Standards Authority (DSA) is an excellent example of the standardisation approach by GDS. It is a body created and dedicated to "put in place a common core infrastructure to join up government and fix the foundations with which it operates". [15] One of the many relevant projects overseen by the DSA is the API Technical and Data Standards [16] which attempts to regulate how APIs should be built.

The Office for National Statistics (ONS) recently started private beta phase of their Integrated Data Service (IDS). The solution is an example of a data centralisation approach, where they will attempt to host data from various sources in one place for easy access [17].

Her Majesty's Revenue and Customs (HMRC) are leading a multi-year, cross-government project called Single Trade

Window (STW) which aims to deliver a unified customer experience for companies importing to and exporting from the UK. The approach they are taking appears to be limited centralisation[5] of data and services with an attempt to change the law to make the sharing of the data possible [18].

The GDS and the CDDO are jointly working on Government Data eXchange (GDX). This approach combines guidance on how to best use the current legal frameworks and technical know-how with tooling to make data exchange easier [14]. This is in addition to the Data Sharing Governance Framework [19] guidance published earlier this year - an attempt to create a "collective commitment for proactive, simpler and faster data sharing" [19].

The above are just a few prominent headline examples of the considerable effort put into addressing the challenges of data discovery and sharing across the Civil Service. There are many more. Several focus on doing what we tried and failed before, but better and on a bigger scale. Others advocate using newer technologies throwing at the problem Solid Pods, GraphQL, blockchain, and the kitchen sink.

2.3 The federated alternative

Despite the concerted efforts of all the teams working on the wide range of solutions, I do not think they will get us closer to the *find me button* I imagined. At the core of the problem are the federated nature and the scale of the organisation. Any potential solution that does not address these is likely to fail. We have plenty of empirical evidence.

I hypothesised that there is a better, federated alternative. A federated solution to a federated problem. I started this project to look for such a thing, to make the *find me button* a little bit less unlikely proposition. By doing so, I wanted to further the ambition of the Digital Government.

3 A SEARCH FOR A BETTER SEARCH

The problem of information discovery in Internet-connected systems is almost as old as the Internet itself. Early approaches, before the invention of World Wide Web (WWW) around 1990, were distributed. Out of necessity. The information was scattered on multiple nodes of the network. The query had to be sent to the individual databases and the results aggregated before presentation to the user. The techniques are now known as search federation and aggregation, sometimes also referred to as meta search or enterprise search.

Z39.50, SRU/SRW and CQL

The Z39.50 standard for information retrieval [20] is an early example of this approach to data discovery. It originated in the 1970s and defined how a client machine can search multiple databases on a single server using a standard TCP connection [21].

At the time, the organisations most interested in the technology were libraries and museums, and in this niche market, solutions implementing the standard evolved. Because of that, the standard never developed a more general abstraction layer for general information exchange and continues to depend on the client and server to share a small set of well-understood information semantics [21].

The idea was further developed into Search/Retrieve via URL (SRU) [22] and Search/Retrieve Web Service (SRW) [23] at the Library of Congress. Both standards work on the same principle as Z39.50 but use different mechanisms to exchange the messages. SRU relies on Hypertext Transfer Protocol (HTTP) and eXtensible Markup Language (XML) messages while SRW provides a Simple Object Access Protocol (SOAP) interface. Both use Contextual Query Language (CQL) [24] to express the queries. The newer standard application is still mostly limited to libraries and museums.

OpenSearch

In 2005 Jeff Bezos announced OpenSearch developed by Amazon.com's subsidiary A9, a web search company. The release note read "OpenSearch is a collection of technologies, all built on top of popular open standards, to allow content providers to publish their search results in a format suitable for syndication" [25].

The idea was to build on the success on Really Simple Syndication (RSS) and expand the news syndication ideas to search. Major web browsers were quick to adopt the search element of it. OpenSearch is the technology that allows us to switch between any number of search engines in the built-in search functionality of Chrome, Edge, Firefox, Safari and many others.

But this is just a small part of what OpenSearch offers. I was much more interested in its ability to syndicate and aggregate search requests. Unfortunately, it is difficult to ascertain how well adopted this technology is as the typical use cases are within large news and search engine organisations that prefer to keep their code closed.

One notable exception and an example of it in the public domain is from Microsoft. Their Search Server 2008 offered federated search functionality [26]. In addition, they have used OpenSearch in Windows OS from Windows 7 released in 2009 to allow search from within the Windows Explorer [27]. Their SharePoint supports OpenSearch too in what Microsoft calls Enterprise Search [28].

3.1 Data warehousing

An alternative approach for information retrieval and exploration emerged in the 1980s[6]. In 1988 IBM researchers published "An architecture for a business and information retrieval system" where they coined the term *business data warehouse*.

It is an example of a centralisation approach. In a process known as Extract, Transform, Load (ETL), information otherwise distributed through the network for operational purposes is moved to a single, central location for analysis and reporting.

The process consists of three phases. *Extraction* - first, the data is read from the operational system. *Transformation* - the data is then transformed into a model more suitable for the analytical processes. *Load* - the transformed data is loaded into central storage where it is indexed and ready for querying.

5. affecting organisations involved in trade

6. the term was first defined in the 1970s, but most of the development and commercial availability happened in the 1980s

ETL often "serves as a central repository to provide search and exploration functionality. ETL is an expensive process that not all organisations can afford and having access to the entire dataset is not possible in many application scenarios, e.g, web sources in the Deep Web or personal data." [29].

The process is intensive as it often requires custom work for each data source to extract and transform it into a data model within which it makes sense when combined with other data sets. Often, the target analytical model is designed to answer a specific set of questions, which means more development is needed when the question set changes.

Most of the data solutions currently used in Civil Service follow this pattern or one of its derivatives like data lakes.

3.2 Crawl-and-index search

The invention of the WWW, a hypermedia-connected network of information, allowed for the 1990s revolution in search. Federation appeared no longer necessary. Neither was the laborious data warehousing and ETL. The information[7] was now publicly available in a standard format. Anybody could start with one web page and explore - crawl the web to index its content in one central place. An ETL process for the web.

Initially, it was done by hand. Sir Tim Berners-Lee maintained the first Internet catalogue - *World Wide Web Virtual Library* - until 1993, but as the number of connected servers grew, it became increasingly difficult to keep up. After 1993 Archie, short for Archives, took over. It was the first automated web crawler and a website that allowed to search the Internet by file name [30].

The idea is much simpler than the federated approach. This type of search consists of three elements. First is a web crawler. Its task is to navigate pages following hypermedia embedded in the documents. A ranking algorithm is then used to index and rank the relevance of each page to a possible search term. Finally, the user inputs a search term which is used to select the most relevant pages from the index.

By the end of the 1990s, web indexing was big business. The crawling and indexing became increasingly sophisticated with the development of algorithms like *RankDex* and *PageRank*. By the beginning of the 2000s, Google became synonymous with 'web search'. The Cambridge Dictionary defines verb *google* as *to search for something on the Internet*.

3.3 Winners and loosers

A review of papers from the 1990s and early 2000s shows that both search approaches were considered a potential solution for the growing problem of discovering the ever-growing information on the Internet.

The librarians, especially in Australia and the USA, were further developing the federated approaches that allowed the public to search their otherwise closed and protected catalogues. At the same time, many companies and researchers were looking at indexing all the publicly available information.

It wasn't obvious which solution would be better. In both cases the challenge was the scale. In 1993, there were 130 websites; in 1998 we already had 2,410,067 of them; and by 2003, there were 40,912,332 websites online [31]. Would we be able to store in one place an index covering all those resources? Would we have the bandwidth necessary to federate each search to thousands, if not millions of servers?

Drivers and inhibitors control the adoption of any technology. Both the centralised and federated approaches had their pros and cons.

The crawling required the data to be publically available and indexed separately to where it is stored, causing a delay between the update of the information and the indexing. Naturally, that created a lag for what was discoverable, but those potentially out-of-date results were available within milliseconds.

Each federated search took longer as it required a time-consuming cascade of network requests to be processed. However, it offered search in restricted datasets and immediate access to the changes.

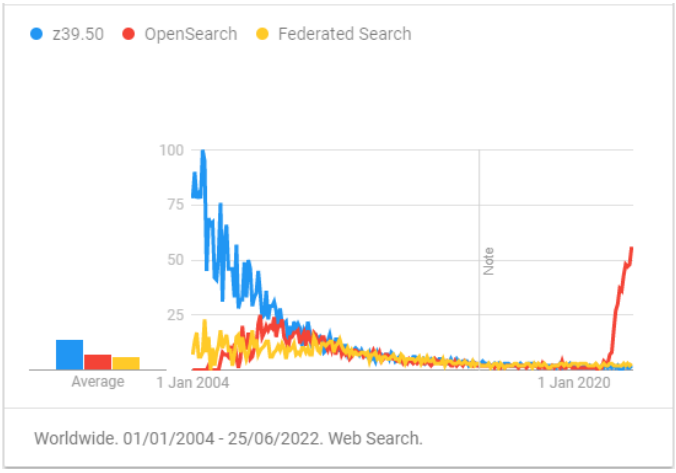

Figure 2. Google trends of topics relating to federated search

In 2006 Apache Hadoop, a set of open-source utilities to process data at a massive scale by distributing workloads, became available. In 2007 we got to 121,000,000 websites. The same year the interest in OpenSearch launched just two years earlier appears to have picked and started declining. As Google Trends shows - Figure 2 - the federated search technologies became less popular[8] over time.

Of course the Google Trends shows only the popular engagement with a concept over time. We can observe the more academic interest by looking at the number of papers covering or mentioning the concept over time. Figure 3 shows a similar decline post-2007 in the number of papers published by year.

It looked like the crawling won the search argument, and the federation gradually faded into irrelevance. Commercial solutions using the latter approach were few and far between. My hypothesis was that federated solutions to federated problems are better. Was I wrong?

I turned to open source projects hosted at GitHub. I have reviewed 76 projects relating to federated search and 31 projects specifically implementing OpenSearch. Their activity

7. at least some of it

8. the sudden rise of OpenSearch in recent years is caused by Amazon repurposing the name for their full-text search service

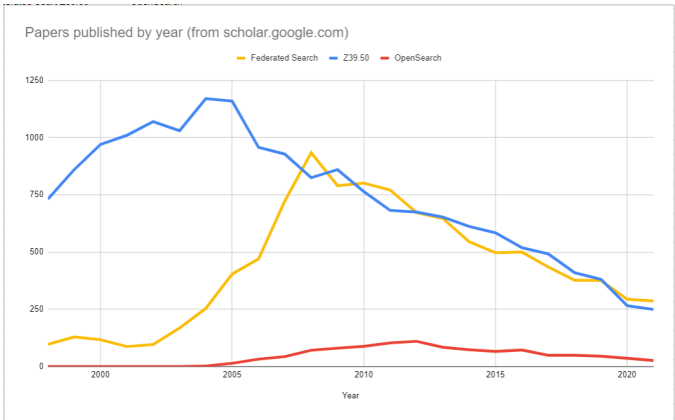

Figure 3. Papers published by year

showed a familiar pattern. They have been losing momentum since the second half of the 2010s — most completely stopped development by 2016.

After interviewing several maintainers of those now abandoned projects, another pattern started emerging. They stopped because they thought the technology had become irrelevant. Some chose to switch to full-text search solutions like ElasticSearch, available since 2010. Others chose GraphQL released in 2015.

4 Making it, somehow, work

I was not ready to admit that my hypothesis was wrong. I turned my interest at the hypermedia in web Application Programming Interface (API).

In his *Architectural Styles and the Design of Network-based Software Architectures* Roy Fielding defined Representational State Transfer (REST) as an "architectural style for distributed hypermedia systems" [32].

This type of architecture might be applicable to the problem of information discovery in the Civil Service because it has been designed to answer problems in "large scale distributed systems, where complete knowledge of all servers would be prohibitively expensive" [32].

Fielding writes about *Internet-scale* systems but not only in the sense of geographical dispersion. "The Internet is about interconnecting information networks across multiple organisational boundaries. Suppliers of information services must be able to cope with the demands of anarchic scalability and the independent deployment of software components" [32].

If not for the word *Internet*, the above describes well the challenges of the Civil Service. We have information networks across multiple organisational boundaries with plenty of anarchic behaviour despite the best efforts to standardise and centralise our efforts.

Unfortunately, the Service Manual has been mandating the use of RESTful web APIs and the API First Design of services for some time now [33]. If this was the solution, we shouldn't have the problem we have!

To understand the issues better, I have spoken to fellow Digital, Data and Technology (DDaT) professionals from across the government. They all, same as us, the Department for International Trade (DIT), claim to follow the guidance. Why does it not work?

On closer inspection the mystery disappeared. Following the 'best' industry practice [34] we ignored the Hypermedia as the Engine of Application State (HATEOAS) in REST. Fielding included it as one of four essential constraints of the architectural style he described. An API not using hypermedia as the engine of its application state is not RESTful at all. In his own words: "If the engine of application state (and hence the API) is not being driven by hypertext, then it cannot be RESTful and cannot be a REST API" [35].

My extensive research has not revealed a single publicly available government API driving its state by hypermedia. Most do not focus on resources and resource manipulation. Despite what we tell ourselves, we do not do REST. Not in the way defined by Fielding. Our 'RESTful' APIs are screaming RPC - Remote Procedure Call - a very different solution to a different problem.

We see HATEOAS as unnecessary. It is awkward to implement. It is not apparent what immediate benefits it offers. The common examples[9] of universal clients for browsing APIs are not compelling enough. When compared to common development practices and standards like OpenAPI they convince nobody to seriously consider the technology.

The examples are often well prepared and presented but fail to convince. Is it because the REST is an architectural style intended for Internet-scale system [32]? Is it because it is a software design on the scale of decades [38]? Is it because it is not common to think at this intended scale when developing yet another microservice?

4.1 Hypermedia in the *found me page*

I have already planned to include hypermedia in the project. The initial idea was to use it only as a feedback mechanism. The *find me button* federates the search. The search network finds and aggregates the information and then presents it to the user as a consolidated view of what the government knows about them. I hoped this would be more convenient than the usual list of search results we know from popular search engines. It could look like Figure 4 - the *found me page*.

The data quality issues quickly become evident. Results come from hundreds of databases maintained by tens of departments, and so it is likely that we would see multiple addresses listed as in the example. Hypermedia embedded in the response would allow the user to click a 'set as current' button to notify all the departments that hold the incorrect data. Single-click right to rectification under UK GDPR [12].

Initially, I intended the hypermedia to be there for the users' convinence only. But could it also address the broader issues described earlier? After all, Fielding's REST including the hypermedia was always intended to solve information exchange problems in large, decentralised systems.

4.2 Hypermedia in REST

If it only was not so difficult! What does it mean to have hypermedia as the engine of application state? If a service exposes an API in REST style, it should be possible to use the service without any prior knowledge, except for the initial Uniform Resource Identifier (URI) and some shared understanding of media types. "From that point on, all

9. Mike Amundsen's books are good source of such examples [36][37]

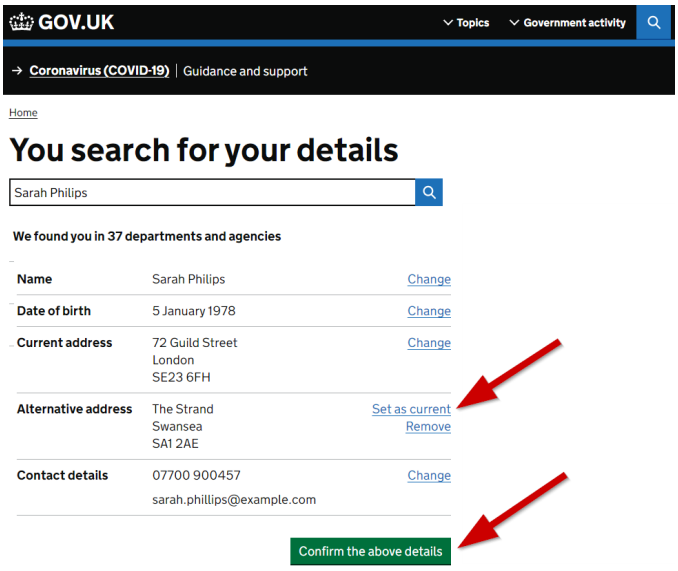

Figure 4. The *found me page* mock-up

application state transitions must be driven by client selection of server-provided choices that are present in the received representations or implied by the user's manipulation of those representation" [35].

This approach affords flexibility. There is loose coupling between the client and the server, as the server is free to control and change its namespace without breaking the clients' functionality. It depends only on both sides agreeing and respecting the shared media type.

The Z39.50 protocol described earlier ultimately failed to address the wider information discovery problem despite decades of research because of close coupling.

First, it was coupled to the Open Systems Interconnection (OSI) framework and, more specifically, to "certain relatively esoteric presentation layer services" at the time when it was already apparent that OSI had failed [21]. Time was lost to move to TCP/IP.

Secondly, the protocol depends on both sides of the exchange sharing the "semantics of the database" which resulted in difficulty in evolving the design and meant it applied only to a relatively confined problem of searching library catalogues [21].

Reading about the history of Z39.50 is difficult not to see REST as an answer to the challenges faced by that much older standard. Perhaps it could help overcome the challenges faced by federated search too?

4.3 Do Internet-scale solutions have to be difficult?

A popular argument against using HATEOAS is how difficult it is to do and how little is gained in return. StackOverflow is full of examples. While this is not the most academic of arguments, to have a chance of success with my federated search ideas, this is the type of audience I will have to convince that HATEOAS is worth a try.

Discussions with software engineers, data and API practitioners in the department and across the government, lead me to believe that the main problem is a lack of practical, applicable examples. Those readily available are trivial and address simpler problems to which alternative solutions already exist. The most common response I get when discussing the subject could be paraphrased as: it sounds like a great idea, I just cannot think of a project in which it would be beneficial to use.

Is it because we focus on building individual, small[10], micro services overlooking the fact that collectively we are designing a federated system across hundreds of organisational boundaries and, that we are, in fact, building an Internet-scale system?

Or is it something else? New, not widely adopted technologies are often complex to implement. Often we have no tooling, examples, and expertise. It is true for REST with hypermedia too, despite the two decades since its definition [39]. However, as those things develop, the technology adoption will become more accessible to the point where we no longer even think about it.

The HATEOAS tooling is slowly becoming available. In 2011 *Collection+JSON* and *HAL* appeared. In 2012 *Hydra* and *Siren* formats were proposed. *JSON:API* followed a year later. These, mostly draft formats, are still evolving but already are offering a varied and increasing number of hypermedia factors[11]. Tooling is still in development but is more available now than before.

Whatever the reason we agree that data and information processing problems at a large scale are technologically and ethically challenging. In both cases, it is worth considering not only what is easy but also what is right to do.

4.4 Hypermedia in wider perspective

Common problems with hypermedia used in APIs include slowness induced by the message size and the need to open individual connections per request in HTTP [38]. The lack of common language to help understand the data and the operations supported by the service are also mentioned [36].

The limited bandwidth problem

The speed problem was a real issue in the 1990s. The way the HTTP worked meant that for every request a new connection had to be negotiated between the client and the server. The handshake contributed significantly to the response time. The problem was mostly solved in 1997 with the specification of HTTP/1.1 [40] which included persistent - by default - connections. But still, APIs requiring multiple roundtrips to the server were considered slow as the client had to wait for the response before issuing the next request. Many of the HATEOAS standards I mentioned before have features specifically designed to address those issues by allowing to remove properties from the response or proactively include extra information to save the round trip [38].

In 2015 the HTTP/2 [41] was introduced that addressed further the problem of time and the bandwidth necessary to service a request. HTTP/2 is no longer a text protocol. It means people cannot easily read the exchange, but it can be more optimised for transfer. Moreover, it offers concurrent queries on a single connection, header caching, and server

10. most projects are small compared to the scale of Internet
11. describing what kind of actions can be performed by a hypermedia driven interaction

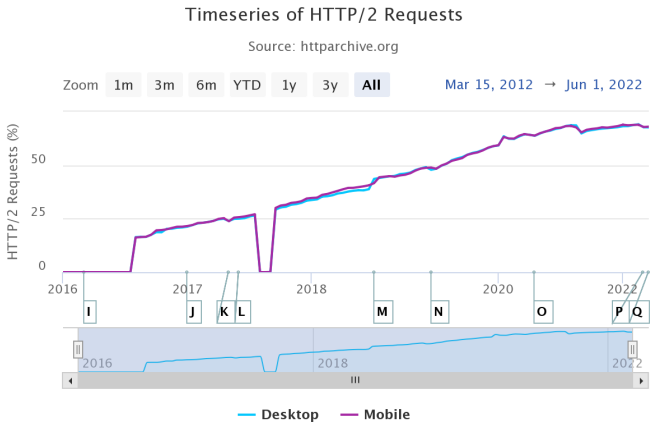

Figure 5. HTTP/2 proportion in requests to *httparchive.org*

data push all features offering greater speeds with less bandwidth utilisation.

If it was not enough the HTTP/3, a proposed standard [42], has just been published (June 2022) and offers even greater speed gains. It will no longer use connection oriented Transmission Control Protocol (TCP). Instead it will utilise QUIC protocol [43] over User Datagram Protocol (UDP) - a connection less transport. According to *caniuse.com* it is already supported by 72% of browsers!

The evolution of HTTP not only makes HATEOAS easier[12] but it also makes the federated approach more likely to return results in a time acceptable to the users.

The problem of sharing just the media type

How can we build two processes, a client and a server, that communicate with each other with nothing but the entry point and the media type known to the client ahead of time?

First, we must consider what a media type is in this context? Formerly known as Multipurpose Internet Mail Extension (MIME) type [44][45], a media type is a data format used to exchange information on the Internet. Hyper Text Markup Language (HTML) is, perhaps, the best example of a media type that allows for hypermedia-driven state transitions.

In a familiar experience, we can navigate the web to find pages on any subject using generic browsers[13] that have no understanding of the content itself. Instead, they understand and interpret the media type rendering the available interactions to the screen as hyperlinks, forms and buttons. It is up to us to decide which interaction to initiate, how to transition the state.

It is possible because of our shared understanding of the world which is not part of the media type and the web server to web browser communication.

Unfortunately, HTML suffers from many other problems which make it rather inconvenient, and in practice, never used media type for API development [36]. Furthermore, even if it was possible to use it in the context of a service-to-service interaction, how can we ensure that the services share the same understanding of information and interactions without creating the close coupling the REST promises to remove?

One possible solution is to add more semantic meaning to the media type. The Semantic Web or Web 3.0 is an example of how HTML evolved to help machine processing of web page information [46].

An alternative is to design media types better suited to service-to-service communication and find ways to describe the semantics of the interaction outside of the medium itself.

Some media types depend on some shared understanding of the domain. Hypertext Application Language (HAL) focuses on two concepts: *links* and *resources* and their names, represented often as simple words. They have to be shared by the client and the server.

Listing 1 shows an example of a HAL response. The client needs to understand what `ea:basket` or `total` is, but as long as those keywords do not change, the server is free to reorganise its interface, offering already some flexibility in the system [47].

```
{
        "_links": {
            "self": { "href": "/orders/124" },
            "curies": [{
                "name": "ea",
                "href": "https://example.com/{rel}" }],
            "ea:basket": { "href": "/baskets/97213" },
            "ea:customer": { "href": "/customers/12369" }
        },
        "total": 20.00,
        "currency": "GBP",
        "status": "processing"
}
```

Listing 1. A HAL example from *stateless.group*

HAL is an example of a media type that focuses only on the link hypermedia factors. Some media types go further. Hydra consists of Hydra Core Vocabulary and JavaScript Object Notation for Linked Data (JSON-LD). It supports both link and to some extent, control hypermedia factors.

Listing 2 demonstrates how both elements of Hydra improve the information exchange. The `@context` property, part of JSON-LD, provides link to a description of what an *issue* is. The definition is independent of both the client and the server, and so while both sides share a dependency, neither is dependent on the other. Additionally, the *operation* property informs the client that it can remove the entity by invoking the *DELETE* operation - part of the core, shared vocabulary.

JSON-LD proved very helpful in providing a shared understanding of data between services. It has found its domain in Search Engine Optimisation (SEO) where it helps search engines to understand the content of web pages. While it is possible to self-declare an entity and use it as the `@context`, the benefits are increased if the definition is widely shared so many services can agree on what the entities mean and how they are structured. *Schema.org* is an example of an attempt to provide such shared understanding. It is a project started by big search engines to improve search results, but we can use the same or similar techniques to improve data exchange between web services through API.

12. and a range of features of the current standards obsolete

13. mobile app development took the opposite course where often applications are built to serve a single purpose and interact with a single service

```
{
    "@context": "https://example.com/issue.jsonld",
    "@id": "/an-issue/1234",
    "title": "An exemplary issue representation",
    "description": "This issue can be deleted",
    "operation": [{
        "@type": "Operation",
        "method": "DELETE"
    }]
}
```

Listing 2. A Hydra example from *markus-lanthaler.com*

Back to the original question. How can we build a client and a server that communicate by sharing just the media type? "The short answer is to stop trying to keep clients and servers in sync by working out ways to share private types. Instead what is needed is a technique for describing data in a way that is not bound to any internal type, programming language, web framework, or operating system" [47]. We need to use "coarse-grained messages that include metadata that describes not only the data being passed but also the state of the application at the time of the request" [47].

The next steps are to extend those principles to descriptions of available actions, followed by metadata about reasons and conditions of use. For now, I call them *Linked Actions* and *Linked Constraints* respectively.

The ideas, independently, are not novel. The Application-Level Profile Semantics (ALPS) is a draft media type proposed by Michael Amundsen in 2014. It provides a way to describe the semantics of an application or a service in a way which is independent of the service itself. The separation affords the descriptions to be shared between services regardless of their specific implementation.

In 2016 the Personal Data Exchange programme at GDS explored "a new model of personal data sharing called Aquae (Attributes, Questions, Answers and Eligibility). It allows public sector services to reuse personal data from around government whilst maintaining the privacy of data subjects, allowing data providers to retain control of data they hold, minimise what is shared and to audit when and how data was used [48].

4.5 So, can we make it work?

All the mentioned formats for *Linked Actions*, *Linked Constraints* and *Linked Data* already exist and at least JSON-LD is often used. However, combined, those three can offer a way to create RESTful systems where clients can use the services provided by the servers without knowing anything except the URI and the media format.

Currently, we approach the our distributed data challenges by doing what we know well. But what if we listened to Martha Lane Fox. What if we tried a revolution instead of evolution [2].

My research identified well-understood but not well-known, not often used technologies. Those so far ignored solutions, especially when combined with previously discussed recent developments in HTTP, might be able to offer a different perspective and get us closer to where we want to be. I think we can make it work, but first...

4.6 The elephant in the room

I spent a lot of time and effort researching how we can discover non-publicly available data across organisational boundaries, all to find a way to make the *find me button* a little less unlikely.

Federated search, together with improvements to HTTP and good RESTful API design, could help. However, often when I make the argument, I am reminded that all of it would not be necessary if not for how the government stores the information about us. I will address it now.

I have already mentioned the revolutionary work of Sir Tim Berners-Lee - the hypermedia-connected web pages. This work significantly contributed, in my opinion, to the near obsolescence of federated search. Now, his more recent project - Solid[14] Pods - might make experiments completely unnecessary.

Solid project "aims to radically change the way web applications work today, resulting in true data ownership as well as improved privacy" [49]. We cannot easily find *our* information the government holds. And it is *our* information. Personal data is *personal* under GDPR. The question I have been asking is how we can make it easier to find *our* data in *government* archives. However, we can ask a different question. Why are *they* holding *our* data in the first place?

Of course, the answer is that they need our data to provide us with their services. However, the technology proposed by Solid team offers an alternative solution. We could be holding our data and letting others, including the government departments, to use it when they need it.

This arrangement has many potential benefits, but what is most relevant in the context of my *find me button* example is that we would not need to look for all the copies of our data and rectify discrepancies. Instead, we would hold the only authoritative copy of the data and be free to update it at will.

This might be the future. There are already examples of the technology starting to work for the citizens. The Flanders Government, in collaboration with Ghent University, are working on *My Citizen Profile* service [50]. Using Solid Pods they plan to offer all 6 million citizens a personal data pod by the end of the year. British Broadcasting Corporation (BBC) are experimenting with using the same technology to free media consumer preferences [51]. The National Health Service (NHS) are working on improving patients' access to their medical records [52], and the GDS are considering using it for a single government login [53].

However, even if successful, even if we adopts this approach, the transformation will not be quick. The departments will store data for years to come. At the same time, there will always be non-personal data that the government stores and may want to discover to help with cross-governmental collaboration. For non-public data discovery, the federated search may still prove a helpful, worthwhile approach.

Simply put, while Solid Pods are the future, they are not a reason not to explore federated search and service discovery right now.

14. nothing to do with SOLID software development principles

5 The ODIS

I started my search for a better search with a clear focus on data discoverability in distributed, non-public systems. However, soon I realised that the potential solution I was investigating was able to answer the wider service-level problem too. Suppose we can use *Linked Data* to solve the data problem in APIs across organisational boundaries. In that case, we should be able to use *Linked Actions* and HATEOAS to solve the service interactions and perhaps, in the future, also address the legal, ethical and operational concerns with *Linked Constraints*.

The challenge is that currently, we do not use the *Linked Data* in this context often. ALPS and Aquae are standards proposed but not adopted so, the *Linked Actions* and *Linked Constraints* are not yet proven. For those ideas to succeed, we need working examples to demonstrate, experiment and discuss the technology.

I have started the ODIS project to validate the proposed solutions and foster the discussion. It is open and publicly available[15]. The name stands for Open, Distributed Information Sharing System.

Currently, the *system* consists of just a few elements, but the name emphasises the idea of bringing together several existing solutions to address the problem.

A demonstration data service holds various sample datasets available through a simple API. In addition, there is an implementation of Open Distributed Information Sharing System (ODIS) node, a service which can be used to construct search networks in various topologies through which searches can be federated. The ODIS nodes use HATEOAS and so allow for experimentation on various media types in RESTful service-to-service interactions.

In the future, ODIS will also include a standard definition and a complete reference implementation. However, so far, the main effort has been on working with the hypermedia types and the ALPS documents to prove that *Linked Actions* can be successfully used to abstract the knowledge about the interactions from both the server and the client.

5.1 ALPS and the ODIS network

Before I explain how the search works in an ODIS network, I will first explain how the network is created following REST principles and taking advantage of ALPS described earlier.

Imagine two services called *A* and *B* managed by two individuals - Alice and Bob. The services are not connected but follow the ODIS protocol. Alice would like to allow her users to benefit from the services offered by the service *B*. She initiates the connection by issuing the instruction to her server *A* to connect to service *B* by giving it the *uri* she got from Bob - *https://b.example.com/odis*.

The services begin the communication shown in Figure 6. *A* initiates a request and the media type negotiation begins. Current, demo implementation settles on Web Service Transition Language (WeSTL)[16] [54] in JavaScript Object Notation (JSON) format. *B* responds with its service description

15. https://github.com/michalporeba/odis

16. while it is convenient, it is only intended as an intermediary representation, from which the responses will be converted into more standard, widely adopted formats like *Hydra* or *JSON:API*

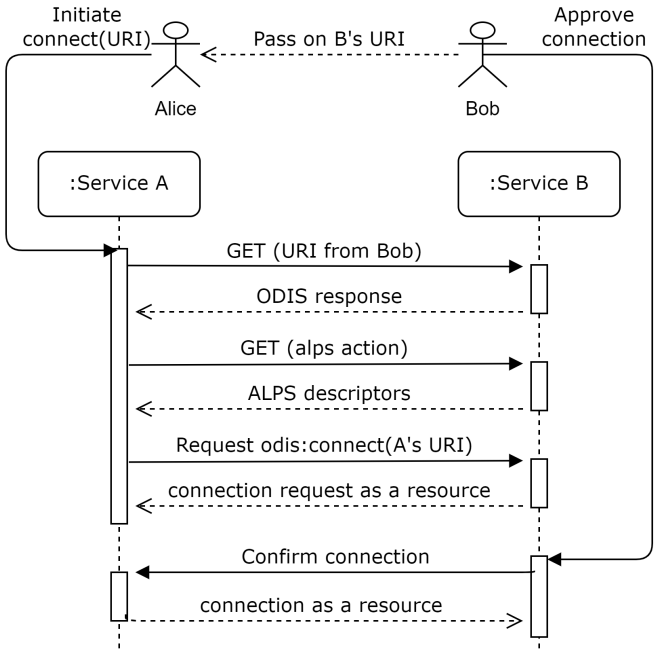

Figure 6. The ODIS connection protocol flow

including link to the `alps` action. The response looks similar to the one in Listing 3.

The `alps` property returned in the first response from *B* is the only direct dependency[17] between *A* and *B*.

Service *A* follows the `alps` link to discover a *profile* like the one in Listing 4 describing what operations are possible. The *B* supports the `odis:itconnect` which *A* needs, and so *A* can take the next step following the connect action. Following the profile *A* passes its own URI when making the request and the *B* returns a connection request resource.

Bob now has to approve the connection request, and when he does, it is *B* that calls *A* to confirm the connection. Connection now has been established, but only from *A* to *B*, which means that *A* can make use of services available at *B* but not the other way round.

```
{ "wstl": {
    "curies": [
    { "name":"o", "href":"https://alps.gov.uk/odis/{rel}"},
    { "name":"l", "href": "http://local.test/odis/{ref}"}
    ],
    "actions": [
    { "name":"self", "type": "safe", "url": "o:" },
    { "name":"alps", "type": "safe", "url": "o:alps" },
    { "name":"o:connect", "type": "safe", "url": "l:connect" },
    { "name":"o:search", "type": "safe", "url": "l:search" }
    ],
    "title": "ODIS - Hello"
}}
```

Listing 3. A simplified ODIS response in WeSTL

Once the connection is established, users with access to *A* can also access[18] services exposed by *B*. For example, when a user requests the `odis:search` operation on *A*, the service will

17. this violates HATEOAS principle and will be improved

18. there will be a layer of authorisation built in

check all the services it knows about and pass on the request to all those which have approved and active connection and which support that action - in this case to *B*.

```
{"alps": {
    "version": "1.0",
    "title": "Open Distributed Information Sharing (ODIS)",
    "doc": {
        "type": "html",
        "href": "https://github.com/michalporeba/odis/"
    },
    "curies": [
    { "name":"a", "href":"https://alps.gov.uk/service/{rel}" },
    { "name":"odis", "href":"https://alps.gov.uk/odis/{rel}"},
    { "name":"s", "href":"https://schema.gov.uk/{rel}"}
    ],
    "descriptor": [
        { "id": "self", "type": "safe"},
        { "id": "home", "type": "safe" },
        { "href": "s:describe", "type": "safe" },
        { "href": "odis:connect", "type": "unsafe",
            "descriptor": { "gs:id": "url" }
        },
        { "href": "odis:node", "type": "safe"},
        { "href": "odis:search", "type": "safe"}
    ]
}}
```

Listing 4. A simplified ALPS profile for ODIS

6.1 Evaluation

The above exchange demonstrates a successful application of ALPS and HATEOAS[19]. In other experiments, I demonstrated with the code developed how such exchange can survive changes to the API without changes to the client or service-to-service interactions.

Based on the experiments and feedback I received, I developed a comprehensive use case illustrating the benefits of a network like ODIS which can extend services available to the users while being agnostic to both the services it brokers and the data it connects. It shows "how a Customer Relationship Managment (CRM) type of service could benefit from the distribution and application of hypermedia in service interactions design" [55].

The project sparked many discussions, connections and new ideas. As a result, I have identified several problem areas that could benefit from solutions based on the ideas already developed and demonstrated by ODIS, but also, potentially, from the extension of work done for data semantics into the semantics of actions and constraints.

To get there, we will need to make working with this technology easier by developing tools for designers, developers and decision-makers. We will need to construct shared but decentralised ontologies to maintain flexibility of louse-coupling of services and not to simply shift tight-coupling from one place to another. In addition, we will need to develop more sample implementations proving and establishing the concepts.

19. with the exception of the single `alps` property

I have identified several projects in our department and across the government that could benefit from decentralised data and service exchange like ODIS. However, before we get there, we must address one critical question. Is the solution we are discussing needed at all? And this time, I am not talking about Solid Pods and future changes to how we store the data!

During one of many debates on APIs and hypermedia somebody said to me: We have thousands of APIs across the Civil Service, and none of them uses HATEOAS or follows your[20] ideas. Isn't this evidence that it is not needed?

It is a fair question that needs addressing, but I am not ready to categorically answer it just yet. More work is needed here. Nevertheless, the possible counter is this: Doesn't the fact that we are still debating how to solve problems in our APIs suggest that we might have been doing it wrong, and so, is it not time to try something new?

At its core, the question is whether the Civil Service systems are Internet-scale or not, whether they should be built to evolve over decades and whether we are prepared to pay the price for it. The financial aspect of the question is essential here as solutions like what I am proposing are often "directly opposed to short-term efficiency" [35].

Working on the ideas presented in this paper over the last 11 months, I have shown that my proposal is a viable solution. Next, we need to establish if it is the *right* solution and see if we can make it simple enough to make it affordable both financially and technically.

6.2 Future development

We must not forget that, for all intents and purposes, the argument I am making has already failed. The term REST is used not for what it was intended. Two decades later it describes any JSON over HTTP communication [35].

At the beginning of this project, I focused on the federated search and simply wanted to use hypermedia and ALPS in my API as a technical, implementation detail to make it work for me. Now, I realise that this *technical detail* is essential for the federated approach to work. The alternatives, briefly mentioned before, like *GraphQL* or full-text search engines are good solutions, but for different problems.

The next steps for ODIS are to switch focus from just the *Linked Actions* and ALPS. We will have to develop specific examples to demonstrate the benefits of all three linked ontologies independently and when combined.

To make more autonomous, hypermedia-driven communication across organisational boundaries on Civil-Service-scale, we will need to agree on how we deal with knowledge about the world between the services. We will have to do so without agreeing on specific shared terms and their meaning or a single source of authority. While we should start with an existing ontology like *schema.org* or *dbpedia.org*, we will likely have to create *schema.gov.uk* and use the richness of relevant standards to manage relationships between the concepts and terms. The standard will have to be constructed through collaboration rather than centralisation, and it will have to be decentralised. Each department must be free to extend and override the shared, agreed model to suit their needs.

20. it is Fielding's idea, but I have been talking hypermedia a lot in some forums recently

7 Conclusions

In my research for this project, I came across an IBC conference paper which read: "The work described in this paper was motivated by the observation that data is not being done well insofar as industry-standard practices do not reflect the public service values" [51]. It resonated a lot with my work.

While what I described above was motivated by much more personal ambition, it inevitably led to a similar conclusion. Perhaps, as a public service, we should not always follow *industry standard practices*. Those, ultimately, often serve different purposes and sometimes are driven by a different set of values.

The work and the research I have done were transformational for my own understanding of information and services in distributed, decentralised systems. It changed my understanding of the unique public service perspective of such systems. The experimental code I have produced has been applied to good effect. Using it, I was able to start, as planned, a discussion about potential alternatives to how we deal with sharing data and services across organisational boundaries in Civil Service.

What might be missing is a clear conclusion of the project. There is still so much to be done. In fact, there appears to be more to do now than when I first started. Nevertheless, I think this is okay.

In 1804 Meriwether Lewis and William Clark embarked from Illinois, US, on a search for a continuous waterway to the Pacific[21]. They were sent to find the Northwest Passage. They failed. Nevertheless, their expedition was a great success, and the list of their achievements is long.

In my search for a better search, I have discovered more questions than answers, but I am confident that what I have learned along the way will help us get closer to that *find me button* and to realising the Digital Government ambition. Perhaps, it has done so already.

Either way, this is not the end. The paper is a progress report. The search for a better search continues.

21. I have learnt about their expedition when reading Designing the Search Experience [56] a book relevant to the research question

REFERENCES

[1] Cabinet Office, "A GDS Story 2010," 2021. [Online]. Available: https://gds.blog.gov.uk/story-2010/

[2] M. Lane-Fox, "Directgov 2010 and beyond: Revolution not evolution," 2010.

[3] Cabinet Office, "Digital Strategy 2012," 2012. [Online]. Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/296336/Government_Digital_Stratetegy_-_November_2012.pdf

[4] OECD, "Rethinking e-Government Services," Tech. Rep., 2008.

[5] Cabinet Office, "UK Digital Strategy," 2017. [Online]. Available: https://www.gov.uk/

[6] OECD, "Digital Government Index," Tech. Rep., 2019.

[7] Government Digital Service, "The next steps for digital, data and technology in government - Government Digital Service," 2021. [Online]. Available: https://gds.blog.gov.uk/2021/04/06/the-next-steps-for-digital-data-and-technology-in-government/

[8] B. Inmon and M. Levins, "Evolution to the Data Lakehouse," 2021. [Online]. Available: https://databricks.com/blog/2021/05/19/evolution-to-the-data-lakehouse.html

[9] K. Schwab, *Shaping the Future of the Fourth Industrial Revolution: A Guide to Building a Better World*. Portfolio Penguin, 2011.

[10] M. Fawler, "Data Mesh Principles and Logical Architecture," 2020. [Online]. Available: https://martinfowler.com/articles/data-mesh-principles.html

[11] "Data Protection Act 2018," 2018. [Online]. Available: https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted

[12] "Individual rights." [Online]. Available: https://ico.org.uk/

[13] Cabinet Office, "Departments, agencies and public bodies." [Online]. Available: https://www.gov.uk/government/organisations

[14] ——, "A GDS Story 2010," 2022. [Online]. Available: https://www.gov.uk/government/publications/roadmap-for-digital-and-data-2022-to-2025/transforming-for-a-digital-future-2022-to-2025-roadmap-for-digital-and-data/

[15] Rosalie Marshall, "Introducing the Government Data Standards Authority," 2020. [Online]. Available: https://dataingovernment.blog.gov.uk/2020/08/27/introducing-the-government-data-standards-authority/

[16] Government Digital Service, "API technical and data standards (v2 - 2019)." [Online]. Available: https://www.gov.uk/guidance/gds-api-technical-and-data-standards

[17] Office for National Statistics, "ONS launches Integrated Data Service to boost government collaboration on data sharing," 2021. [Online]. Available: https://www.ons.gov.uk/news/

[18] Cabinet Office, "The UK Single Trade Window Consultation Paper (draft)," 2022.

[19] Central Digital and Data Office, "Data Sharing Governance Framework," 2022. [Online]. Available: https://www.gov.uk/government/publications/data-sharing-governance-framework/data-sharing-governance-framework

[20] "Information Retrieval (Z39.50): Application Service Definition and Protocol Specification," 2003. [Online]. Available: https://www.loc.gov/z3950/agency/Z39-50-2003.pdf

[21] C. A. Lynch, "The Z39.50 Information Retrieval Standard," 1997. [Online]. Available: https://www.dlib.org/dlib/april97/04lynch.html

[22] "Search/Retrieval via URL - standard," 2016. [Online]. Available: https://www.loc.gov/standards/sru

[23] "Search/Retrieval Web Service - standard," 2016. [Online]. Available: https://www.loc.gov/standards/sru/companionSpecs/srw.html

[24] "Contextual Query Language - specification," 2016. [Online]. Available: https://www.loc.gov/standards/sru/cql/spec.html

[25] "Bezos on "open source search"," 2016. [Online]. Available: https://blogs.publicradio.org/futuretense/2005/03/amazons-jeff-be.html

[26] Microsoft, "Federated Search Overview." [Online]. Available: https://docs.microsoft.com/

[27] ——, "Getting Started with Federated Search in Windows - Win32 apps." [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/search/getting-started-with-federated-search-in-windows

[28] ——, "Enterprise Search." [Online]. Available: https://docs.microsoft.com/

[29] D. Collarana, *Strategies and Techniques for Federated Semantic Knowledge Integration and Retrieval*, ser. Studies on the Semantic Web. IOS Press, 2020.

[30] A. Wall, "History of Search Engines: From 1945 to Google Today," 2016. [Online]. Available: http://www.searchenginehistory.com/

[31] [Online]. Available: www.internetlivestats.com

[32] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[33] Governmet Digital Service, "Service Manual." [Online]. Available: https://www.gov.uk/service-manual

[34] Members of the BCS Internet specialist group, "Hypermedia controls in REST - The final hurdle," 2017. [Online]. Available: https://www.bcs.org/articles-opinion-and-research/hypermedia-controls-in-rest-the-final-hurdle/

[35] R. T. Fielding, "REST APIs must be hypertext-driven," 2008. [Online]. Available: https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

[36] M. Amundsen, *Building Hypermedia APIs with HTML5 and Node: Creating Evolvable Hypermedia Applications*. O'Reilly Media, Inc., 2011.

[37] ——, *RESTful Web Clients*. O'Reilly Media, Inc., 2017. [Online]. Available: https://www.oreilly.com/library/view/restful-web-clients/9781491921890/

[38] D. Beattie, "The Rest of ReST," 2016. [Online]. Available: https://youtu.be/g8E1B7rTZBI

[39] H. Vu, T. Fertig, and P. Braun, "Verification of hypermedia characteristic of restful finite-state machines," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 1881–1886. [Online]. Available: https://doi.org/10.1145/3184558.3191656

[40] R. T. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," 1997. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2068

[41] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7540

[42] M. Bishop, "HTTP/3," 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9114

[43] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9000

[44] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," 1996. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2046.html

[45] "Media Types." [Online]. Available: https://www.iana.org/assignments/media-types/media-types.xhtml

[46] D. Allemang, J. Hendler, and F. Gandon, *Semantic Web for the Working Ontologist*. Association for Computing Machinery (ACM) BOoks, 2020.

[47] M. Amundsen, *Design and Build Great Web APIs*, ser. The Pragmatic Programmers. The Pragmatic Bookshelf, 2020.

[48] S. W. Andy Bennett, "Aquae: Personal data for cross-Government services," 2019. [Online]. Available: https://www.register-dynamics.co.uk/data-trusts/aquae-model.html

[49] "Original page of the Solid project." [Online]. Available: https://solid.mit.edu/

[50] "Flemish government allocates 7 M€ funding for first 2 years of SolidLab Vlaanderen." [Online]. Available: https://www.ugent.be/ea/idlab/en/news-events/news/flemish-government-7m-eur-funding-solidlab-vlaanderen.htm

[51] E. S. et al., "ENHANCING MEDIA THROUGH THE DEVELOPMENT OF A PUBLIC SERVICE DATA ECOSYSTEM," in *IBC2021 TECH PAPERS: CUTTING EDGE TECHNOLOGIES - A PREVIEW OF SOME EXPERIMENTAL CONCEPTS*, 2016.

[52] "This Month in Solid - October 2020," 2020. [Online]. Available: https://solidproject.org/newsletter/2020-10-01

[53] B. Glick, "UK government turns to Tim Berners-Lee startup for digital identity plan," 2021. [Online]. Available: https://www.computerweekly.com/news/252506983/UK-government-turns-to-Tim-Berners-Lee-startup-for-digital-identity-plan

[54] M. Amundsen, "WeSTL - Web Service Transition Language - specification," 2016. [Online]. Available: https://rwcbook.github.io/wstl-spec/

[55] M. Poręba, "Open Distributed Information Sharing (ODIS)," 2022. [Online]. Available: https://github.com/michalporeba/odis

[56] T. Russell-Rose and T. Tate, *Designing The Search Experience: The Information Architecture Of Discovery*. Morgan Kaufmann, 2013.