A Solid-based Architecture for Decentralised Interoperable Location Data

Maxim Van de Wynckel*, Beat Signer

Web & Information Systems Engineering Lab, Vrije Universiteit Brussel, 1050 Brussels, Belgium

Abstract

In today's technological world of privacy-conscious users, the tracking of individuals via different positioning systems and services can be considered obtrusive. Furthermore, linking and integrating data from these positioning systems is not always possible or requires the major effort of creating new interfaces between systems. In this paper, we propose an architecture for the realisation of a decentralised positioning system based on the W3C's Solid platform specification. Using this specification, sensor data as well as an individual's location information is stored in secure decentralised data stores called Pods, that are hosted by user-selected Pod providers. We demonstrate that these Pods do not only offer transparent and interoperable data stores for persisting sensor data as well as processed location information, but also aid in linking multiple positioning systems for high- and low-level sensor fusion. For indoor positioning, this interoperability provides a way to offer users a single location-based service while also providing additional semantic context for other positioning systems to improve their data output. Developers of indoor positioning systems can store all data in a format that is readable, understandable and accessible by any other system that their users might be using, enabling collaboration between researchers and companies implementing these indoor positioning systems.

Keywords

location-based services, decentralised data

1. Introduction

With current positioning systems that are used both indoors and outdoors, data is usually stored in a database that is under control of a specific application. Neither users nor external developers always have access to the raw data unless a dedicated interface enables its extraction. Even with the ability to extract the data of a positioning system, the interoperability between multiple positioning systems and applications remains a difficult task due to different data formats, the lack of documentation or missing information on how the data has been processed. In an indoor positioning system (IPS), the additional information might include the system setup with the used sensors and their features.

In several user studies on location privacy conducted before 2010 [1, 2], it was concluded that the majority of people do not put a lot of value in their location privacy due to their ignorance of the possible implications. In recent years, users are becoming more conscious about their

IPIN 2022 WiP Proceedings, September 5 - 7, 2022, Beijing, China

*Corresponding author.

mvdewync@vub.be (M. Van de Wynckel); bsigner@vub.be (B. Signer)

https://wise.vub.ac.be/maxim-van-de-wynckel (M. Van de Wynckel); https://beatsigner.com (B. Signer)

0000-0003-0314-7107 (M. Van de Wynckel); 0000-0001-9916-0837 (B. Signer)

CEUR Workshop Proceedings (CEUR-WS.org)

privacy on the Web and on smartphone applications [3]. For indoor positioning systems that are often implemented and deployed for individual buildings, this could cause users to distrust the use of a system or application that is able to track their location. Our proposed architecture provides users transparency about their location data as well as any system that can access that data, and allows them to define individual access rights for each system or application. By defining the use of a common vocabulary for the location and sensor data, we are able to create an architecture that can be implemented to seamlessly integrate with other positioning systems or applications without the need for additional interfaces, increasing a user's trust that no other systems can have unauthorised access to their location.

In 2016, the W3C Solid Community Group led by Tim Berners-Lee introduced *Solid (Social Linked Data)* [4], a platform for decentralised social applications based on linked data. Solid stores a user's data and information in secure data vaults called *Pods* that are independent from the applications that create, edit or process the data. Users can choose the Pod provider and control the access rights of applications for certain subsets of their data vault. Stored data can range from binary media files to structured linked data that is described by documented vocabularies and specifications. We propose the use of Solid for creating a positioning system that stores all its data, ranging from sensor data to high-level computed positions as linked data in a user-owned Pod store. We show how this storage makes IPS-generated data interoperable between other positioning systems, how privacy can be preserved and how consumers of location data can benefit from Solid.

Finally, we showcase the implementation of our proposed Solid-based architecture via some non-trivial example positioning systems. In this demonstration we highlight how different systems can access a user's location data after they have given their consent, how this data is semantically described using existing standards and recommendations and how the semantics helps in processing the data in a meaningful way.

2. Related Work

Decentralising a positioning system is not a new concept, but has mostly relied on proprietary decentralised parts with the focus of decentralising the processing rather than the decentralisation of data to preserve user privacy. Talat et al. [5] proposed a decentralised approach to store privacy-preserving trajectory data using decentralised blockchains. Their motivation was that current enterprise systems are built on the notion of trust by being in control of the data and the inability to provide users data access.

Existing localisation systems that preserve the privacy of tracked users apply location privacy preservation mechanisms (LPPMs) such as obfuscation, anonymisation or cryptography [6]. These techniques often focus on the anonymisation of a location-based service (LBS) while we aim to create a data exchange of all information relevant for high- and low-level sensor fusion between multiple systems.

LBS interoperability has been conceptualised in various research [7, 8, 9] with a main focus on the LBS as a centralised middleware that handles the reasoning, privacy and querying of data. The reasoning over data is an important aspect, as it enables the LBS to use all the context that is available to provide a better single location output. To enable this reasoning, the spaces

and deployments of the positioning systems that provide data to these services is one of the key areas that aforementioned research focuses on.

A prototype for a decentralised LBS using the Geolocation API has been realised by the Solid community [10]. Thereby, the Geolocation API outputs the position at multiple levels of granularity, such that a user can decide which accuracy can be shared with which applications. While the prototype offers historical location data, different levels of granularity and a vocabulary based on other specifications, it lacks additional semantics that would allow its use with multiple positioning systems. Further, the stored data cannot be expanded with domain-specific data.

With our approach, we demonstrate how the decentralised Solid platform can act as an interoperability layer for a location-based service, as well as a layer for low-level sensor data and information regarding building infrastructure. Instead of modifying the data with LPPMs, we aim for privacy by transparency with different types of access for different granularities of data, similar to the prototype by Stratsianis [10]. Different than the work of Talat et al. [5], Solid allows users to stay in control of their data that can be accessed by applications.

3. Solid-based Architecture

Our architecture for creating a decentralised positioning system is built on top of the Solid project by Tim Berners-Lee [4]. Solid offers decentralised data vaults called *Pods*, storing a user's data. Users can choose the *provider* storing their Pod and thereby decide which organisation to trust storing their data. Applications and websites that want to store or fetch a user's data can ask the user to authenticate themselves to their Pod provider, giving those applications access to parts of the vault where they can create or read data.

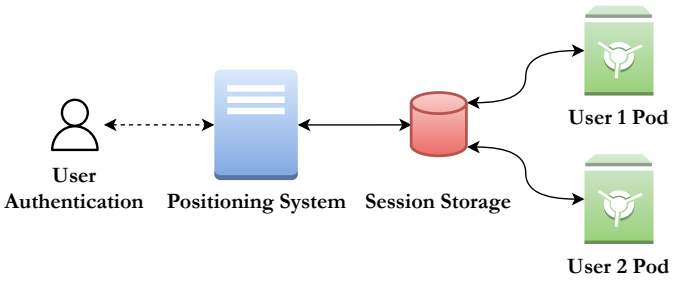

Figure 1: Basic architecture of linking a positioning system with Solid

In Figure 1 we show the basic architecture of a positioning system making use of Solid. Users authenticate through a user interface (i.e. either a website or a smartphone application). The authentication process in this system provides a session key that can be used to access the user's private Pod. Each user will have their own application-specific unique session key stored in the session database, enabling access to their Pod. These keys are the only data that has to be stored locally by the system. In Section 3.2 we provide practical examples on how an indoor positioning system would perform this authentication in a real-life scenario.

Once the positioning system has access to a user's private Pod, the Pod will be used to store all relevant tracking data. This might range from processed data such as a user's geographical position to raw sensor data that should be persisted. While the positioning system can still store

some, if not all, data on privately owned servers, the philosophy of Solid is that this would be governed by laws such as the European Digital Markets Act (DMA) [11]. Similar to the governing of the usage of data vaults, applications with read access to a Pod can theoretically clone the data locally. However, laws like the European General Data Protection Regulation (GDPR) [12] aim to protect users from such practises.

The user has control and transparency on the data that is being created or updated by a positioning system, as well as the ability to revoke access to this data at any time. Compared to the three-layer model of an LBS [13], our approach moves the responsibility of the reasoning middleware layer to the application that reads the data, but with additional semantic rules and knowledge on how this reasoning can be performed. This offers a lot of flexibility, making the decentralised vault not simply an LBS, but also a service for raw unprocessed data.

3.1. Interoperability

Location data should be interoperable between different systems, meaning that data created by an indoor positioning system in one building should be compatible with another positioning system or application. This allows our positioning data to be processed by different positioning systems to enable the handover of tracking [14], or allow multiple indoor positioning systems to be linked in one non-proprietary smartphone application. Interoperability on this scale further enables the decoupling of positioning systems from the user interfaces and applications that generate and use the data.

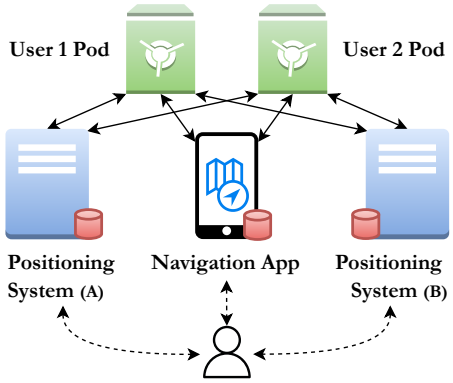

Figure 2: Two positioning systems linked to the same user pods

Figure 2 shows an architecture where a user authenticates to two individual positioning systems that work independently from each other. Both systems can access the same data vaults. In addition, we have a navigation application that uses the data stored in the decentralised Pods as a location-based service. Alternatively, a single application could be responsible for obtaining sensor data on a smartphone that is then being processed by a positioning system.

Solid allows the storage of regular documents such as media files as well as semantic linked data [15]. Semantic data is machine readable and understandable structured data based on vocabularies and rules that semantically describe the meaning and relations of data. The data is represented in *triples* consisting of a *subject*, a *predicate* and an *object*. Each subject, predicate or object can be identified by a Unique Resource Identifier (URI) that might be used to define

other predicates describing the URI. Further, concepts can be defined in multiple documents on the Web, creating a semantic web of data.

We will formalise a collection of vocabularies that cover the general setup of a positioning system and its algorithms, the output data with different granularities and a vocabulary that provides additional context on the accuracy of each output position in Section 3.3.

3.2. Authentication and Access Control

In order that an application can make use of Solid, the user needs to authenticate the application to the Pod provider. For an indoor positioning system, we provide two different ways how this authentication would be done by a user. In the first scenario, the user uses their smartphone to authenticate the positioning application to the Solid provider. This can be through a smartphone application or online website that handles the positioning. The access credentials might be stored locally on the smartphone or in the browser cache if no server authentication is required.

If the positioning system obtains sensor data on a mobile device but processes the data on a server, this server requires access as well. In such a scenario, user authentication is still achieved through a web user interface. After a successful authentication, the access credentials are stored on the server that can execute actions on behalf of the user.

Both types of authentication require an informed user consent [16] where the user is aware what application will access their Solid Pod, what location data they require access to and optionally what they are going to do with the location data. For positioning systems that do not use an Internet-connected device, such as positioning systems that track the position using other hardware (e.g. multi-camera multi-person tracking or RFID-based tracking), the system should still authenticate the user. We envision this type of authentication at the time when the hardware is handed to the person or when the user provides consent for being tracked.

Authentication can be revoked or downgraded at any time, preventing a positioning system or application from accessing and updating a user's position. Each user is uniquely identified through a *WebID* URI (e.g. `http://ipin2022.solidweb.org/profile/card#me`) and every WebID directs to the user's Pod that is classified via different paths depending on the type of data. Every WebID directs to a public profile providing a basic overview of the user's profile information. Other paths such as `/properties/position.ttl` are custom *datasets* added by applications.

Access control within Solid is handled on a dataset level. Each container (i.e. path in the URI) and file can have specific access control policies (ACP) for each authenticated application, user groups or everybody. There are five different roles of access a user or application can have to a dataset as shown in Table 1. The three main ACP modes include *write* access with the ability to create, update and delete information, *read* access of all data in a dataset and *append* access that can only create new information. Owners and editors are similar despite of the fact that owners can also manage the access control settings to a dataset.

An application using Solid as a location-based service only requires the *visitor* role. A smartphone application generating sensor data only needs to create new data and can therefore use the *submitter* role. A positioning system should have the role of a poster in order that it is able to create new sensor readings and computed location results, but cannot change or modify information generated by other positioning systems. Further, a trusted system that should have the ability to remove or modify incorrect sensor data from other systems requires an *editor* role.

Table 1
Access Control Policies (ACP) roles and modes

	write	read	append	control
owner	✓	✓	✓	✓
editor	✓	✓	✓	✗
poster	✗	✓	✓	✗
submitter	✗	✗	✓	✗
visitor	✗	✓	✗	✗

3.3. Vocabulary

Our collection of vocabularies and concepts was defined based on the common needs of a positioning system, the available techniques and algorithms and the data they produce [17, 18]. This vocabulary is applied in a dataset within a Solid Pod as shown later in Section 3.2. With the vocabulary we should support the following concepts that are required for creating interoperable indoor- and outdoor positioning systems:

- **System description**: The positioning systems that are used should be semantically described with their properties, deployments, algorithms and technologies.
- **Output properties**: Each tracked entity should have a set of specific properties such as a position, orientation, velocity or any other data that a positioning system can provide.
- **Accuracy**: The accuracy of a position or any type of sensor should be specifiable for each output that is provided. While some sensors provide all results with the same assumed accuracy, other processed properties such as a position can have a varying accuracy for each individual observation.
- **Granularity of properties**: The more accurate the location data, the more sensitive the information. Not every system requires access to a high accuracy. Our solution should support different levels of granularity for each property belonging to an entity. This enables a user to limit access to a dataset with a high accuracy.
- **Historical data**: We want to keep track of computed positions. A positioning system could use the historical data to predict future movement and trajectories [19].
- **Raw sensor data**: While our main focus is to store location data, it should be possible to store raw sensor data belonging to a person (e.g. raw accelerometer data) in a similar way as a property of a tracked entity.
- **Spatial reference system**: In order to support the interoperability of spatial sensor data in different environments, a spatial reference system should be provided to determine how the position from different systems relate to each other without requiring them to output the data in the same reference system.

The Open Geospatial Consortium (OGC) together with the W3C created the Sensor, Observation, Sample and Actuator ontology (SOSA) [20]. This ontology is extended with the Semantic Sensor Network Ontology (SSN) [21] that together can describe systems, their sensors, the properties of entities and individual observations for the properties, capabilities of a system and their precision and accuracy [22].

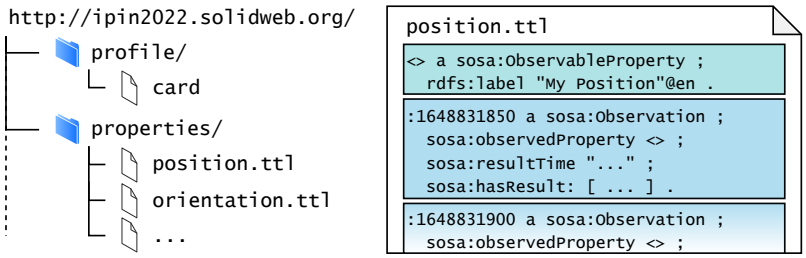

Figure 3: Overview of a fictional user's data stored in a Solid Pod

As a joint specification by the Spatial Data on the Web Working Group (SDWWG) and the W3C, this vocabulary is chosen as the main ontology for describing a positioning system and its inner workings. For observational values, we make use of the Quantities, Units, Dimensions and Data Types ontologies [23] that support and describe most units. Spatial data is represented with GeoSPARQL [24] that conforms with known ISO standards. While we focus on expressing data commonly provided by a positioning system, SOSA offers the ability to describe individual sensors and other raw sensor data that could be of use for other systems.

With the Semantic Web, other domain-specific vocabularies can be linked to the core concepts introduced in this section. This supports the description of deployments using domain-specific vocabularies such as IndoorGML [25] or a more elaborate description of sensors using M3 [26]. To ensure that each application with data access follows the same conventions, shape validation that verifies that the data follows the specifications could be used in the applications [27].

The concept of *observable properties* from the SOSA ontology is used to define the properties of a user that can be observed, such as their position, orientation or velocity. Figure 3 shows an overview of the data in a Pod. We use the profile card to store references to the properties that are available for our user using the `ssn:hasProperty` predicate (see Section 4.1 for an example). Information about each property and the observations is located in a container called `properties/`. A new dataset file is created for each property, containing its semantic description. Individual *observations* of these properties are included in the dataset file. These observations contain the timestamp when the observation was created, the result and optionally additional semantics about the used algorithms and techniques (i.e. the `sosa:Procedure` predicate).

```
@prefix : <http://.../properties/position.ttl#> .

<> a sosa:ObservableProperty ;
  rdfs:label "Geographical Position"@en ;
  ssn:hasProperty :accuracy .

:accuracy a ssn-system:Accuracy ;
  schema:maxValue "5.0"^^xsd:float ;
  schema:unitCode qudt-unit:M .
```

Listing 1: Example position property (`position.ttl`)

In Listing 1 we show the creation of a position property using the combination of the SOSA, SSN, SSN-Systems[1], Schema.org and QUDT ontologies (prefixes redacted from the listing). The position property has a property on its own that specifies its expected maximum accuracy.

[1]https://www.w3.org/TR/vocab-ssn/#System-capabilities

```
1  @prefix profile: <http://.../profile/card#> .
2  @prefix example: <http://purl.org/ipin2022-solid#> .
3
4  :1646891100 a sosa:Observation;
5    sosa:hasFeatureOfInterest profile:me;
6    sosa:usedProcedure example:qrscanner_checkin ;
7    sosa:hasResult
8        [ a geosparql:Geometry;
9          geosparql:asWKT """
10           <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
11           POINT Z(5.893628199 46.641890130 15)
12           """^^geosparql:wktLiteral ;
13         geosparql:coordinateDimension 3 ;
14         geosparql:hasSpatialAccuracy
15             [ a qudt:QuantityValue ;
16               qudt:numericValue 598 ;
17               qudt:unit qudt-unit:CentiM ] ];
18   sosa:observedProperty <> ;
19   sosa:resultTime "2022-03-10T06:45:00.000Z"^^xsd:dateTime .
```

Listing 2: Example position observation (`position.ttl`)

A single observation of this property is shown in Listing 2 with the algorithm that generated the observation on line 6, timestamp on line 19 and the result itself written as a geometry with a well-known text (WKT) representation [28, 29] for the position on lines 9 to 12. This WKT position contains the longitude, latitude and coordinate reference system to express the position. The accuracy of the position as provided by the system that created the observation is described on lines 14 to 17 with a one-dimensional value (distance) in a specified unit. The spatial accuracy can be changed to a 2D area or 3D volume if the system is able to provide this.

3.4. Decentralised Data

The infrastructure and setup of a positioning system can be important to perform decision-level position fusion from different systems. Instead of duplicating this data in each user Pod, we envision that each positioning system contains a dataset describing its layout and setup. Optionally, parts of this data can also be private with access-on-demand, in a similar way as a user's location and sensor data would be private.

In Listing 3 we show a basic example of an indoor positioning system. The system defines the deployment, the location of the deployment and procedures providing semantics on the techniques the positioning system uses. This basic example does not go in depth into the semantics of a procedure and the deployment, but can be expanded with domain-specific ontologies as mentioned in Section 3.3.

3.5. Privacy and Transparency

The privacy aspects of a positioning system cover both the user's privacy when using an indoor positioning system, as well as the privacy of a company or building owner implementing such a system. As a user, one does not want to be tracked by unauthorised applications or positioning systems whereas a company or building owner does not want too many details about the building infrastructure to be revealed.

```
@prefix : <http://purl.org/ipin2022-solid#> .

:system_indoor a ssn:System ;
  rdfs:label "My Indoor Positioning System"@en ;
  ssn:hasDeployment :office ;
  ssn:implements :qrscanner_checkin, :qrscanner_checkout .

:office a ssn:Deployment, geosparql:SpatialObject ;
  vcard:hasAddress [ ... ] ;
  geosparql:hasGeometry [ geosparql:asWKT """
      POLYGON Z ((4.39220 50.82040 9, 4.39227 50.82043 9,
                  4.39224 50.82046 9, 4.39217 50.82043 9))
      """^^geosparql:wktLiteral ] .

:qrscanner_checkin a sosa:Procedure ;
  rdfs:label "QR-scanner Check-in"@en ;
  rdfs:comment "..."@en ;
  ssn:hasInput [ ... ] ;
  ssn:hasOutput [ ... ] .
```

Listing 3: Example indoor positioning system definition

Other than LPPMs [6] that modify or encrypt the data to ensure privacy, the privacy in our architecture mainly focuses on the transparency and control of granularity in which applications can access a user's location. In our vocabulary we define properties for an entity that we are tracking. We consider the positioning system that has read and write access to a particular property as a trusted entity for managing that property. To control the level of granularity, multiple properties can exist for the same type of measurement (e.g. position or orientation), each with their own accuracy, update frequency or limited information. By supporting these different levels of granularity, the user can control which application(s) have access to what accuracy of their position, similar to the proof-of-concept LBS of Stratsianis [10].

3.6. Decentralised Processing

Since a Pod is a decentralised storage of individual observations by a positioning system or even a sensor, it can be used as a broker for the decentralised processing of observation data. One part of the positioning system adds the observations to the Pod. Another part of the positioning system reads these observations to continue processing them. Solid supports live notifications [4] where an open connection is made to a Pod dataset to listen for changes. This allows a positioning system to check for new observations appended to a property.

3.7. Fusion

With multiple positioning systems having access to create observations of a position, there is a need to control how the position is updated. In our vocabulary presented earlier in Section 3.3, we mentioned the use of the SOSA ontology to create individual observations for each result. Multiple positioning systems can create multiple results or observation on the same property (e.g. position) with additional semantics that indicate *how* and *who* made the observation. Using this information, querying the data allows for specifying this additional semantics. To ensure that a positioning system can only write new observations but not delete any, our access control settings can be configured to ensure that only new observations can be appended.

Reasoning on the context is an existing aspect in much of the related work that offers interoperability of location-based services [7, 8, 9]. In the aforementioned related work, the service is a centralised server that reasons on the semantic data it is given and provides the application with a processed position that makes use of this reasoned data using the algorithms and techniques determined by the LBS. With our approach, we want to offer more control over the reasoning on the provided high-level location data.

In our linked data approach, fusion relies on semantic reasoning [30] and the ability to perform complex queries on the data. A semantic reasoner provides additional information depending on rules, axioms and constraints defined in the vocabulary. This enables queries to use a larger set of inferred information to perform queries on. A basic example is an axiom specifying that a particular predicate (e.g. `ssn:hasProperty`) has an inverse (e.g. `ssn:isPropertyOf`). With this rule, data that only uses the `ssn:isPropertyOf` predicate can still be queried with its inverse. Such a query is executed using the *SPARQL* query language and can be performed either locally on the complete dataset obtained from Solid, or on the server side (selective read operation) based on a Solid server implementation.

4. Implementation

To demonstrate how Solid can benefit the indoor positioning community, we created a non-trivial implementation using a basic IPS, one outdoor positioning system and a consumer application that shows a user's recent data along with additional information. Using this implementation, we want to demonstrate how a user's personal position, orientation and velocity is stored, how applications gain access to this data and how our solution might benefit tracked users.

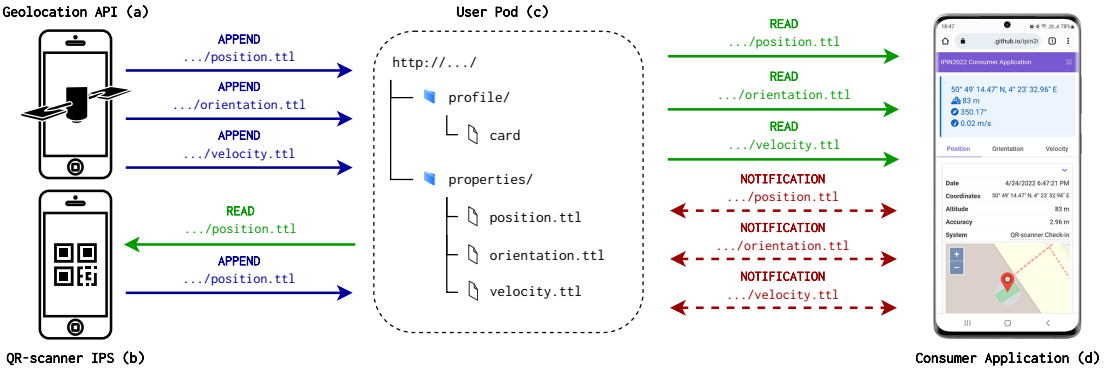

Figure 4: Demonstrator with two positioning systems (a, b) and one consumer application (d) connected to the same user pod (c)

Figure 4 shows the technical setup of our demonstrator. For the scope of this demonstration, the used indoor positioning system is a QR code *check-in* system that will update the position of a user depending on the check-in and check-out scanning (see Figure 4b). The outdoor positioning system uses the Geolocation API [31] to update the position, orientation and velocity as depicted in Figure 4a.

While in this demonstration we focus on positioning data, the SOSA ontology enables the modelling of properties and observations of any sensor data that belongs to the tracked entity. Properties such as relative positions, raw signal strengths from beacons or raw IMU sensor data can be modelled similar as shown in Listing 2 with a different value for `sosa:hasResult`. Finally, in Section 4.3 we demonstrate how another positioning system or application can use the semantic context of the two positioning systems to predict a more reliable output.

4.1. Properties

In our demonstration we manage three properties of the entity that we are tracking; the position, orientation and velocity. Our Geolocation API provides all three properties, while the QR scanning only updates the position property. For the scope of this demonstration, the speed and orientation is expressed as a one-dimensional value in the `sosa:hasResult` of the example observation in Listing 2.

Properties can be discovered by reading or querying the profile card of a Solid Pod[2]. Listing 4 shows an example profile card describing a person and `sosa:FeatureOfInterest` which are accessible URIs to the datasets containing the observations of these properties.

```
:me a schema:Person, sosa:FeatureOfInterest, foaf:Person ;
  vcard:bday "1995-03-10"^^xsd:date ;
  vcard:fn "Maxim Van de Wynckel" ;
  vcard:hasAddress [ ... ] ;
  vcard:organization-name "Vrije Universiteit Brussel" ;
  ssn:hasProperty </properties/orientation.ttl>,
    </properties/position.ttl>, </properties/velocity.ttl> ;
```

Listing 4: Example profile card with properties

All position observations from every positioning system are stored in the same dataset (`*.ttl` file[3]). These timestamped observations include information on the system that observed them and the procedures used by that system. In the case of the IPS, it also includes a reference to the room where a check-in occurred.

4.2. Applications

Figure 4a shows the Geolocation API application that only appends new observations to the user pod. As shown in Section 3.2, this type of system only requires the *submitter* role. The QR scanner application determines a check-in or check-out depending on the current position (i.e. if the last position is inside a room, scanning the QR code means the user leaves it) and therefore requires read permissions. The minimum required role for this system is a *poster*. For the application that visualises the data, we only require the *visitor* role with read permission.

Figure 5 shows the authentication and visualisation of the data in the Pod. A user starts by selecting their Pod issuer after which they are redirected to the provider's login page to request access. After access has been granted, the application can query the last positions, velocities and orientations. Additional information that is referenced within the observations, such as

[2]http://ipin2022.solidweb.org/profile/card
[3]http://ipin2022.solidweb.org/properties/position.ttl

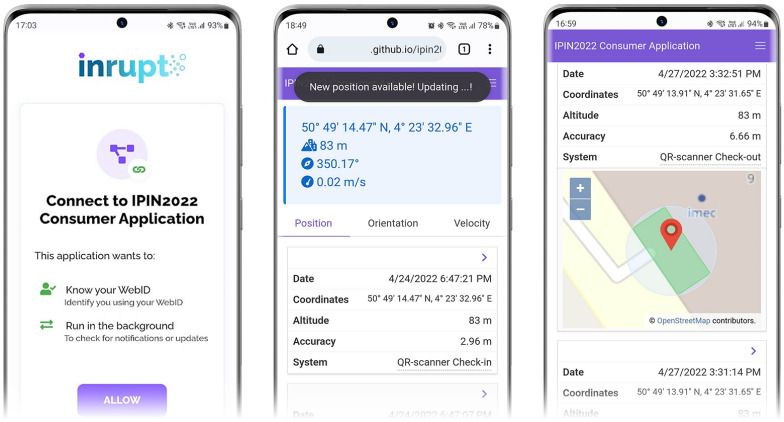

Figure 5: Login authentication example and consumer app

the room, will be fetched from the public triple dataset semantically explaining the shapes and locations of the rooms[4]. Users of a Solid-based application only need to remember a single login, which benefits the use of an IPS that does not require an additional account.

We provide an open source version of all applications along with more supplemental material on GitHub [32]. In this supplemental material, we deliver instructions on how to set up the applications and view the data. Our applications were developed using the OpenHPS framework [33] as a new module named @openhps/solid[5].

4.3. Querying

As SPARQL is not officially required by the Solid specification due to its computational complexity on the server, not every provider supports it. In our example implementation, we use the Comunica [34] SPARQL engine that supports local queries on fetched datasets as well as server-side querying on supported Solid providers. The engine supports link traversal-based query execution [35] where the set of documents will continuously expand by following the URIs within the dataset that one performs the query on. This allows us to perform queries without previous knowledge of the datasets.

Listing 5 shows the SPARQL query used by the consumer application to obtain the last 20 positions, the time when they were observed and the accuracy in metres. With this example we want to highlight the advantage of additional semantic context and linked data provided by the used vocabularies. Our query is executed on the dataset of the profile where the properties are referenced. The property URIs mentioned in the profile card link to other datasets that contain the observations that we aim to query. These datasets linked within the profile will be automatically fetched by using link traversal. A GeoSPARQL 1.1 query function is used for converting the well-known text literal to a GeoJSON format (line 9) in order to immediately use the output in our application. We also use additional semantics from the QUDT ontology to convert the accuracy to a base unit regardless on the unit used in the data (lines 12 to 16).

[4]https://purl.org/ipin2022-solid
[5]https://github.com/OpenHPS/openhps-solid

```
SELECT ?posGeoJSON ?datetime ?accuracy {
  ?profile a sosa:FeatureOfInterest ;
           ssn:hasProperty ?property .
  ?observation sosa:hasResult ?result ;
               sosa:observedProperty ?property ;
               sosa:resultTime ?datetime .
  ?result geosparql:hasSpatialAccuracy ?spatialAccuracy ;
          geosparql:asWKT ?posWKT .
  BIND(geof:asGeoJSON(?posWKT) AS ?posGeoJSON)
  ?spatialAccuracy qudt:numericValue ?value ;
                   qudt:unit ?unit .
  OPTIONAL { ?unit qudt:conversionMultiplier ?multiplier }
  OPTIONAL { ?unit qudt:conversionOffset ?offset }
  BIND(COALESCE(?multiplier, 1) as ?multiplier) # Default 1
  BIND(COALESCE(?offset, 0) as ?offset) # Default 0
  BIND(((?value * ?multiplier) + ?offset) AS ?accuracy)
} ORDER BY DESC(?datetime) LIMIT 20
```

Listing 5: SPARQL query on the profile card dataset

As shown in Figure 4, the consumer application listens for update notifications on the properties. These are used to obtain live updates of the position. In a positioning system this could be used as push-based source for sensor and location data. With our example implementation in Figure 5, we visually showcase these position changes as a pop-up notification.

5. Conclusions and Future Work

We presented a new Solid-based architecture enabling the creation of a decentralised positioning system where user-owned location and sensor data is stored in a user's data vaults. This decentralised storage allows the users to remain in control of their data, while allowing different positioning systems and applications to access and use the data given their consent.

We realised a basic implementation of the proposed architecture where two positioning systems and one location consumer application store and read data from the same decentralised vault. The source code and additional supplemental material is publicly available on GitHub [32].

While Solid is still under development, it highlights the need and advantages to decouple a user's data from applications that produce and consume the data. With location and sensor data being such a sensitive property of a person, it highly benefits to be stored in the decentralised vaults provided by Solid. We showcased that this change in storage is not only ensuring transparency, but the interoperability of data can help in linking multiple indoor positioning systems and improve the reasoning performed on the data. By using linked data in Solid, we might also leverage the data vaults as communication brokers for transmitting sensor observations from one part of a positioning system to another part by using live update notifications [4].

Future work should expand the vocabulary of SOSA and SSN to further semantically indicate what algorithms and techniques are being used, the different relevant types of observable properties that can be provided by a positioning system and a more universal way to identify the quality of a property. With such a vocabulary, more practical examples can be created for demonstrating high- and low-level sensor fusion. Companies might also store the information about their buildings in decentralised Pods using additional vocabularies such as IndoorGML [25], in order to make this information publicly available.

References

[1] D. Cvrcek, M. Kumpost, V. Matyas, G. Danezis, A Study on the Value of Location Privacy, in: Proc. of WPES 2006, 2006. doi:doi.org/10.1145/1179601.1179621.

[2] J. Krumm, A Survey of Computational Location Privacy, Personal and Ubiquitous Computing 13 (2009). doi:10.1007/s00779-008-0212-5.

[3] C. Pilton, S. Faily, J. Henriksen-Bulmer, Evaluating Privacy-determining User Privacy Expectations on the Web, Computers & Security 105 (2021). doi:10.1016/j.cose.2021.102241.

[4] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, T. Berners-Lee, Solid: A Platform for Decentralized Social Applications Based on Linked Data, Technical Report, MIT CSAIL & QCRI, 2016.

[5] R. Talat, M. S. Obaidat, M. Muzammal, A. H. Sodhro, Z. Luo, S. Pirbhulal, A Decentralised Approach to Privacy Preserving Trajectory Mining, Future Generation Computer Systems 102 (2020). doi:10.1016/j.future.2019.07.068.

[6] B. Liu, W. Zhou, T. Zhu, L. Gao, Y. Xiang, Location Privacy and Its Applications: A Systematic Study, IEEE Access 6 (2018). doi:10.1109/ACCESS.2018.2822260.

[7] J.-W. Kim, J.-Y. Kim, C.-S. Kim, Semantic LBS: Ontological Approach for Enhancing Interoperability in Location Based Services, in: Proc. of OTM 2006, 2006. doi:10.1007/11915034_103.

[8] S. Ilarri, A. lllarramendi, E. Mena, A. Sheth, Semantics in Location-based Services, IEEE Internet Computing 15 (2011). doi:10.1109/MIC.2011.156.

[9] K. Lee, J. Lee, M.-P. Kwan, Location-based Service Using Ontology-based Semantic Queries: A Study With a Focus on Indoor Activities in a University Context, Computers, Environment and Urban Systems 62 (2017). doi:10.1016/j.compenvurbsys.2016.10.009.

[10] S. Stratsianis, SOLID Project: Location Data in Solid (Linked Data), https://github.com/SharonStrats/SOLIDLBSPrototype, 2021.

[11] L. Cabral, J. Haucap, G. Parker, G. Petropoulos, T. Valletti, M. Alstyne, The EU Digital Markets Act: A Report From a Panel of Economic Experts, Publications Office of the European Office, 2021. doi:10.2760/139337.

[12] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation), Official Journal of the European Union 59 (2016).

[13] J. Schiller, A. Voisard, Location-based Services, Morgan Kaufmann, 2004.

[14] R. Hansen, R. Wind, C. S. Jensen, B. Thomsen, Seamless Indoor/Outdoor Positioning Handover for Location-based Services in Streamspin, in: Proc. of MDM 2009, 2009. doi:10.1109/MDM.2009.39.

[15] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American 284 (2001).

[16] S. Bu-Pasha, A. Alen-Savikko, J. Makinen, R. Guinness, P. Korpisaari, EU Law Perspectives on Location Data Privacy in Smartphones and Informed Consent for Transparency, European Data Protection Law Review 2 (2016). doi:10.21552/EDPL/2016/3/7.

[17] H. Liu, H. Darabi, P. Banerjee, J. Liu, Survey of Wireless Indoor Positioning Techniques and Systems, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37 (2007). doi:10.1109/TSMCC.2007.905750.

[18] Y. Gu, A. Lo, I. Niemegeers, A Survey of Indoor Positioning Systems for Wireless Personal

Networks, IEEE Communications Surveys & Tutorials 11 (2009). doi:10.1109/SURV.2009.090103.
[19] F. Zampella, A. R. J. Ruiz, F. S. Granja, Indoor Positioning Using Efficient Map Matching, RSS Measurements, and an Improved Motion Model, IEEE Transactions on Vehicular Technology 64 (2015). doi:10.1109/TVT.2015.2391296.
[20] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, M. Lefrançois, SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators, Journal of Web Semantics 56 (2019). doi:10.1016/j.websem.2018.06.003.
[21] K. Janowicz, M. Compton, The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration Into the Semantic Sensor Network Ontology, in: Proc. of SSN, 2010.
[22] M. Compton, et al., The SSN Ontology of the W3C Semantic Sensor Network Incubator Group, Journal of Web Semantics 17 (2012). doi:10.1016/j.websem.2012.05.003.
[23] R. Hodgson, P. J. Keller, J. Hodges, J. Spivak, QUDT-Quantities, Units, Dimensions and Data Types Ontologies, http://qudt.org, 2014.
[24] R. Battle, D. Kolas, GeoSPARQL: Enabling a Geospatial Semantic Web, Semantic Web Journal 3 (2011).
[25] H.-K. Kang, K.-J. Li, A Standard Indoor Spatial Data Model: OGC IndoorGML and Implementation Approaches, ISPRS International Journal of Geo-Information 6 (2017). doi:10.3390/ijgi6040116.
[26] A. Gyrard, S. K. Datta, C. Bonnet, K. Boudaoud, Cross-Domain Internet of Things Application Development: M3 Framework and Evaluation, in: Proc. of FiCloud 2015, 2015. doi:10.1109/FiCloud.2015.10.
[27] R. Zhu, C. Shimizu, S. Stephen, L. Zhou, L. Cai, G. Mai, K. Janowicz, M. Schildhauer, P. Hitzler, SOSA-SHACL: Shapes Constraint for the Sensor, Observation, Sample, and Actuator Ontology, in: Proc. of IJCKG 2021, 2021. doi:10.1145/3502223.3502235.
[28] Information Technology—Database Languages—SQL Multimedia and Application Packages — Part 3: Spatial, Standard ISO/IEC 13249-3:2016, International Organization for Standardization, Geneva, Switzerland, 2016.
[29] Geographic Information: Well-known Text Representation of Coordinate Reference Systems, Standard ISO 19162:2019, International Organization for Standardization, Geneva, Switzerland, 2019.
[30] R. Shearer, B. Motik, I. Horrocks, HermiT: A Highly-Efficient OWL Reasoner, in: Proc. of OWLED 2008, 2008.
[31] A. Popescu, Geolocation API Specification 2nd Edition, 2016.
[32] M. Van de Wynckel, B. Signer, IPIN 2022 Solid Examples, https://github.com/OpenHPS/ipin2022-solid/, 2022.
[33] M. Van de Wynckel, B. Signer, Indoor Positioning Using the OpenHPS Framework, in: Proc. of IPIN 2021, 2021. doi:10.1109/IPIN51156.2021.9662569.
[34] R. Taelman, J. Van Herwegen, M. Vander Sande, R. Verborgh, Comunica: a Modular SPARQL Query Engine for the Web, in: Proc. of ISWC 2018, 2018. doi:10.1007/978-3-030-00668-6_15.
[35] O. Hartig, J.-C. Freytag, Foundations of Traversal Based Query Execution Over Linked Data, in: Proc. of Hypertext 2021, 2012. doi:10.1145/2309996.2310005.