The Punya Platform: Building Mobile Research Apps with Linked Data and Semantic Features

Evan W. Patton[1(✉)], William Van Woensel[2], Oshani Seneviratne[3], Giuseppe Loseto[4], Floriano Scioscia[4], and Lalana Kagal[1]

[1] Massachusetts Institute of Technology, Cambridge, MA 02143, USA
{ewpatton,lkagal}@mit.edu
[2] Dalhousie University, Halifax, NS B3H 4R2, Canada
william.van.woensel@dal.ca
[3] Rensselaer Polytechnic Institute, Troy, NY 12180, USA
senevo@rpi.edu
[4] Polytechnic University of Bari, 70125 Bari, BA, Italy
{giuseppe.loseto,floriano.scioscia}@poliba.it

Abstract. Modern smartphones offer advanced sensing, connectivity, and processing capabilities for data acquisition, processing, and generation: but it can be difficult and costly to develop mobile research apps that leverage these features. Nevertheless, in life sciences and other scientific domains, there often exists a need to develop advanced mobile apps that go beyond simple questionnaires: ranging from sensor data collection and processing to self-management tools for chronic patients in healthcare. We present Punya, an open source, web-based platform based on MIT App Inventor that simplifies building Linked Data-enabled, advanced mobile apps that exploit smartphone capabilities. We posit that its integration with Linked Data facilitates the development of complex application and business rules, communication with heterogeneous online services, and interaction with the Internet of Things (IoT) data sources using the smartphone hardware. To that end, Punya includes an embedded semantic rule engine, integration with GraphQL and SPARQL to access remote graph data, and support for IoT devices using Bluetooth Low Energy and Linked Data Platform Constrained Application Protocol (LDP-CoAP). Moreover, Punya supports generating Linked Data descriptions of collected data. The platform includes built-in tutorials to quickly build apps using these different technologies. In this paper, we present a short discussion of the Punya platform, its current adoption that includes over 500 active users as well as the larger app-building MIT App Inventor community of which it is a part, and future development directions that would greatly benefit Semantic Web and Linked Data application developers as well as researchers who leverage Linked Open Data resources for their research.

Resource: http://punya.mit.edu

Keywords: Mobile app development · Research apps · Linked Data · Semantic rules · Internet of Things · GraphQL · Rapid prototyping

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 563–579, 2021.
https://doi.org/10.1007/978-3-030-88361-4_33

1 Introduction

Scientific research apps rarely leverage the advanced sensing, interaction, and computing capabilities of smartphones. A recent survey [34] found that most smartphone apps in psychological studies do not use smartphone features such as sensors and complex analytical methods. Instead, they are limited to porting existing interfaces to mobile screens to offer mobility (e.g., ad-hoc interactions) via the smartphone. Indeed, it is difficult and time-consuming to integrate an app with peripheral sensors, link to online data sources, and offer complex decision logic. One has to deal with heterogeneous services, protocols and schemas, as well as build complex application and business rules. This results in high software development costs. Since researchers usually rely on one-shot grant funding to develop (and maintain) research apps, and often operate in exploratory settings where app requirements may change over time, they may lack funds to cover costly software development projects [16].

Prior experiences by the authors have demonstrated this: Praino, Scioscia *et al.* [43] developed a mobile patient diary for Systemic Sclerosis patients to annotate the evolution of symptoms via scientifically validated questionnaires, on-device camera, and wearable devices. However, due to difficulty achieving secure mobile communication with electronic health records (EHR), data was not sent: instead, a summary report was generated and communicated to physicians. Moreover, due to heterogeneous and proprietary device protocols, manually customized modules were needed for different types of wearables. Van Woensel *et al.* [55] developed an intelligent mobile diary for Atrial Fibrillation (AFib) patients to enter daily symptoms and vitals, offering local decision support for time-sensitive health feedback. However, a reasoning system had to be manually ported to the device for decision support; and, due to similar reasons as [43], the app could only integrate with a single, off-the-shelf peripheral device.

These experiences highlight the need for an easy to use, open source, platform for developing research apps that provides mechanisms to access peripheral sensors, link to online data sources, and use decision logic.

In addition, we propose that Linked Data (LD) and Semantic Web (SW) technology will help democratize the development of research apps, as it affords creators the following:

- SPARQL [42], GraphQL [19] and Linked Data Platform (LDP) [52] queries and requests, in terms of a domain ontology, to access online data and services; which avoids dealing with discordant schemas and protocols.
- Semantic Web of Things (SWoT) [46] offers methods and tools that generate a unifying layer, in terms of a domain ontology, across heterogeneous IoT device services that hides manufacturer-specific protocols.
- Well-established semantic formalisms, such as OWL [33], SWRL [22], and N3 [5], offer the tools and techniques to define and execute complex processing, application, and decision logic to implement Expert System features.
- Collected and processed research data can be annotated with relevant ontologies and uploaded to a semantic repository or SOLID pod [47].

– Auto-generating wire-frame Linked Data forms from domain ontologies, with fields connected to ontology terms, to avoid boilerplate view-controller code.

In general, relying on re-usable, well-known domain ontologies, standards and tools, avoids re-inventing the wheel in the form of yet another custom data model, online/device service schemas and proprietary protocols; and the downstream efforts required by consumers to support and implement them. The Punya platform [28] is an end-user development environment built on MIT App Inventor [57], a web-based platform that offers a drag-and-drop interface for easily building interfaces and application logic. Punya expands on MIT App Inventor by adding LD & SW affordances to support research apps.

Punya was originally conceived in 2013 as a way to support relief efforts during humanitarian crises, e.g. conflicts in countries as South Sudan, Iraq, and Yemen and the Central African Republic, and Typhoon Haiyan in the Philippines. The goal was to enable relief workers to quickly put together mobile apps, which could be customized to the language/cultural/technical requirements of the specific region and crisis, to aid co-ordination of relief efforts, and to provide information to key decision makers. However, we realized that it was useful for much more than disaster management and started exploring its use in ecology, tracking air pollution, personal health management and other areas.

2 Related Work

Apple ResearchKit [4] is a software framework for apps that let medical researchers gather study data through surveys, forms, and activities, and supports integration with the HealthKit [3] and CareKit [2] frameworks to gather data from peripheral health devices. These 'kits' allow researchers to gather informed consent, create surveys, visualize trends in the data and conduct active evaluations. While ResearchKit is excellent at creating survey-based research apps with customized workflows, it currently lies beyond its purview to connect to existing, online health data sources, or implement complex application and processing logic; nor does it afford the general advances of relying on semantic ontologies. Also, since ResearchKit is rooted in the iOS framework, building an app that uses ResearchKit will require knowledge of the underlying framework—one needs to install the XCode IDE and manually add many code snippets to tie together a ResearchKit app—this increases the learning curve and is not ideal for researchers looking to rapidly build study applications. In contrast, the Punya platform provides a convenient drag-and-drop environment for researchers to quickly prototype their ideas and easily access Linked Data resources through its in-built Semantic Web capabilities.

Node-RED [37] provides a browser-based editor to wire up event-based applications quickly, and it is popular in applications that use IoT components [7]. A Node-RED application can be run on a web server, Raspberry Pi, Arduino, or on an Android device using an emulator. However, Node-RED rather focuses on setting up data flows including IoT devices and online services, as opposed

to the prototyping of general (research) apps, outfitted with a fully-fledged UI and leveraging Linked Data and Semantic Web features as outlined before.

For many types of consumer-facing applications, designers can utilize prototyping tools to ensure high fidelity and adaptability in the final application delivered to their target userbase. For example, Qualtrics [45] is a simple-to-use, web-based survey tool to conduct survey research, evaluations, and other data collection activities. Designers, with no prior experience, are able to use its research suite to build surveys, send surveys and analyze responses. Some other popular prototyping tools include InVision [23], Marvel [31], MockPlus [35], and Proto.io [44]. These tools allow their users to collaborate, research and test their ideas on the cloud-based digital platform, and even integrate with several third-party workflow products and services, where designers can import mockups from Sketch or Photoshop easily. Several of these products have companion apps for mobile platforms such as iOS and Android, enabling designers to create mockups natively on the devices. However, regarding the development of research apps, none of these tools allow going beyond relatively simple survey-based applications, which do not leverage Linked Open Data nor open-source and free-for-unrestricted-use software.

Other frameworks, such as Apache Cordova[1] or React Native[2], focus on web-first principles—i.e., using web technologies for mobile app development. As another example, Progressive Web Apps (PWAs)[3] are single-page web applications that present as native applications, if the mobile platform supports it. These approaches are useful for developing cross-platform apps using a single code base, especially for developers comfortable with JavaScript and web APIs. However, these technologies may be unapproachable for researchers who need to develop apps and who are not trained as software engineers.

3 Punya Components for Semantic Research Apps

The Punya platform builds on MIT App Inventor, providing a suite of additional components that enrich the platform for building semantically-enabled, research-oriented mobile apps. Briefly, the MIT App Inventor interface allows app creators to visually compose one or more screens of an application by dragging and dropping user interface elements into a mock phone screen (called the *Designer*). The behavior of the app is scripted in a visual programming language built on the Google Blockly framework (called the *Blocks Editor*).[4] Apps can be tested in real time using an app called the Companion (Punya provides its own version). Once an app is complete, it can be compiled into a app package (currently Android only) for distribution, such as through the Google Play Store. In this section, we discuss a subset of these components, starting with the core LD support (Sect. 3.1) and then elaborating on Semantic Web of Things, online LD access, and Expert System features (Sects. 3.2–3.4).

[1] https://cordova.apache.org/.
[2] https://reactnative.dev/.
[3] https://web.dev/progressive-web-apps/.
[4] https://developers.google.com/blockly.

3.1 Linked Data Elements

One strength of Punya is its ability to integrate with LD resources as we have previously discussed [28,29]. We briefly summarize each component and discuss how they can be used specifically for building LD-enabled research applications.

- **LinkedData** component wraps a Jena [13] Model object, and, in line with the building block-based design paradigm of App Inventor, offers a block-based interface to an RDF graph, allowing the update and retrieval of RDF data. The component supports reading and writing graph data on the web using SPARQL, and on local storage by serializing the graph as Turtle. An important feature of the `LinkedData` component is its ability to convert the contents of a `LinkedDataForm` into an RDF graph, and, inversely, populating these forms with RDF data. Using this functionality, app developers can use the LinkedData component as a general data store for application content and similarly use the content to drive the user interface.
- **LinkedDataForm** provides the ability to annotate components with `InverseProperty`, `ObjectType`, `PropertyURI`, and `SubjectIdentifier` fields, which can be filled using an autocomplete feature. This allows laying out LD-aware components, similar to HTML `form` elements with RDFa [1]. The contents of the form can then be used to construct RDF graphs based on the form fields and their annotations. `LinkedDataForm`s can be nested to create arbitrarily complex UIs and ensuing RDF graph structures.
- **LinkedDataListPicker** is an LD-enhanced version of the App Inventor `ListPicker` component. It populates its list by evaluating a SPARQL query against a remote endpoint. The labels for the entities are retrieved using the well-known properties *rdfs:label*, *skos:prefLabel*, *foaf:name*, and *dc:title*.
- **Linked Data Form Generator** is a feature of Punya developed by WeiHua Li [29] that allows auto-generating a wire-frame `LinkedDataForm` from a given RDFS or OWL ontology. Researchers can then build an app quickly from best-in-class ontologies for their domain(s) and greatly reduce the amount of manual development.

3.2 Semantic Web of Things

The *Semantic Web of Things* (SWoT) vision [46] integrates knowledge representation and reasoning techniques from the Semantic Web into Internet of Things (IoT) architectures. SWoT enables new classes of smart applications that augment real-world objects, locations, and events with machine-understandable data, annotated with a domain ontology, using mobile and pervasive devices such as smartphones, wearables, and IoT sensors. Currently, Punya supports *Linked Data Platform for the Constrained Application Protocol* (*LDP-CoAP*), a SWoT protocol that enables lightweight, LD-based resource dissemination and discovery in dynamic ad-hoc contexts. Moreover, developers can use App Inventor support for Bluetooth Low Energy (BLE) that allows for low-energy consumption of mobile services.

- **LDP-CoAP Web of Things protocol** is grounded on the LDP W3C Recommendation [52] as reference format and guidelines for managing collections of Linked Data resources on the Web. However, LDP only defines resource management primitives for `HTTP`, leaving out Web of Things (WoT) scenarios where more lightweight application protocols are required. CoAP [9] is an application-level protocol for machine-to-machine (M2M) communications, based on a loosely coupled stateless client/server model. LDP-CoAP [30] defines an adaptation of the LDP specification for CoAP, which allows publishing Linked Data on the WoT while preserving all LDP features and capabilities. Using the `LdpCoapClient` Punya component, a mobile app can expose data, collected via embedded sensors or peripheral devices, as RDF resources to other parties through an LDP-CoAP server. In particular, the component supports the `GET`, `PUT`, `POST` and `DELETE` CoAP methods to request, create, update and remove LDP Resources, which can be organized in hierarchical relationships by means of LDP Containers. To discover RDF resources exposed by external devices, a key feature in dynamic SWoT contexts, the Punya LDP-CoAP component supports the Constrained RESTful Environments (CoRE) Link Format protocol and its discovery functionality [48].
- **Bluetooth Low Energy.** Introduced in 2010, BLE aims to overcome the limitations of the traditional Bluetooth technology in regards to energy consumption and device interoperability. In particular, BLE allows low-energy communication with remote devices (e.g., BLE beacons) in terms of standard-compliant services (e.g., heart rate, blood pressure services). This facilitates the development of smart, energy-efficient mobile apps that monitor personal and contextual data using different types of peripherals and sensors. The App Inventor BLE extension, `BluetoothLE`,[5] provides standards-based Profiles and Services to connect to a range of BLE-enabled peripheral devices, such as health and fitness monitors, environmental sensors, and more. To that end, `BluetoothLE` includes a range of functions for advertising and discovering services, connecting remote devices, and communicating with them.

3.3 Online Data Access

Most use cases in Sect. 4 illustrate the need for integration with remote data sources for research apps. To that end, we have introduced two new features in Punya. The first is support for GraphQL, which can be used to query a number of graph-based data sources (e.g., Facebook) that would otherwise be opaque silos. The second is support for SPARQL in the blocks language, which makes it easier to construct syntactically valid SPARQL queries. We briefly describe each of these features and how they can be used.

- **GraphQL** is a declarative language for querying graph data, originally developed at Facebook and open sourced in 2015 [18]. A number of large companies offer GraphQL endpoints for querying their data. To facilitate unlocking these

[5] http://iot.appinventor.mit.edu/#/bluetoothle/bluetoothleintro.

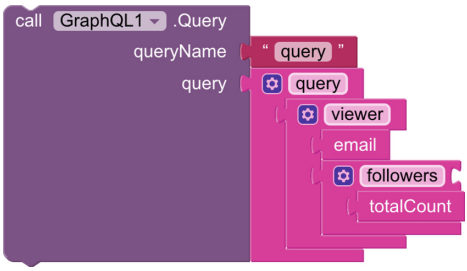

Fig. 1. An example of a GraphQL query for user email and number of followers using the dynamically generated GraphQL blocks.

silos, we have included a GraphQL component [14]. The GraphQL component leverages the introspective nature of the GraphQL language to generate blocks for the types supported by the endpoint. This allows for dynamically building queries while developing the app. GraphQL can be used, for example, to query the social graph of a research study participant, for social research. Figure 1 shows an example of how one might construct a GraphQL query requesting the number of followers of a user via a social network. The blocks are dynamically created by examining the server's schema and shown in context when hovering the cursor over the type names.

– **Writing SPARQL:** Previously, Punya required app developers to programmatically construct SPARQL queries using the built-in text block capabilities. It is possible, for example, to store query templates as file assets in the application. These queries can then be read into memory, and substitutions applied before executing the query against an endpoint. The new version of Punya provides built-in SPARQL blocks inspired by [10]. The SPARQL block functionality performs appropriate type checking on the connections to prevent the construction of syntactically invalid queries. An example SPARQL query to check for drug-drug interactions is shown in Fig. 2.

3.4 Ruleset Construction and Evaluation

Mobile rule engine features allow developers to deploy complex semantic reasoning locally on the smartphone. In general, this can be utilized to realize a form of edge computing [51]—improving response times, reducing bandwidth and need for continuous connectivity, increasing privacy protection and, in some cases, removing the need for online services for storage and processing support.

Of course, important considerations with regards to local processing include performance, scalability, and battery consumption. Patton *et al.* [40] performed a study on battery use during OWL reasoning, comparing battery consumption of 4G, 3G, and WiFi radio. The authors found that it takes less energy to reason on several ontologies using Apache Jena [13] compared to requesting the results from an external source via the 3G or 4G. Although Bobed *et al.* [8] and Kazakov

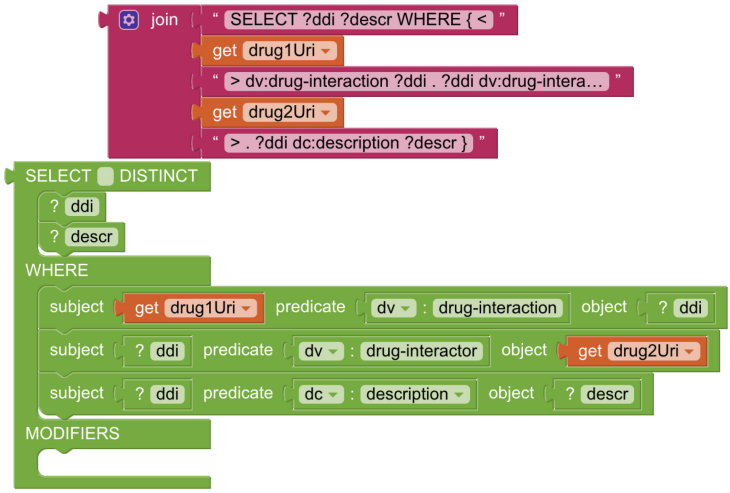

Fig. 2. An example SPARQL query to check DrugBank for drug interactions. Top: Old style; Bottom: New style.

et al. [24] found orders of magnitude difference between reasoning on PC and Android, they also showed promising trends: reasoners on the Android RunTime (ART), which features ahead-of-time compilation, were around two times faster than in Dalvik [8]. In prior work, the same team also found a performance increase of around 30% between Android devices only one-year apart [58]. Van Woensel *et al.* [54] found acceptable performance for Apache Jena on Android for OWL ontologies comprising 500 statements or less. As these studies indicate the feasibility of local reasoning on smartphones, we added to Punya the `Reasoner` component: supporting rule-based reasoning using RDFS, a subset of OWL, or even custom rulesets added by the developers using `Ruleblock`'s.

- **Reasoner** is a component newly added to the Punya platform. It uses the Jena [13] rule-based reasoner to make inferences over the RDF graph provided by a `LinkedData` component. Jena provides built-in rules for RDFS and subsets of OWL, and custom rules can be provided from files or via `RuleBlock`'s (see below). Rule evaluation occurs on a background thread for performance reasons. Reasoning begins when the app invokes the reasoner's `Run` method, and an event, `ReasoningComplete`, is fired when the reasoning has completed and results are available.
- **Writing Rulesets** dynamically based on user input is facilitated by a set of blocks for the language to make syntactically correct rules. These `Ruleblocks` reuse blocks from the SPARQL functionality (see above), e.g., the triple pattern and variable declaration blocks. The `Ruleblocks` provide support to model both forward chaining and backward chaining (Fig. 3). Forward rules can be used to generate backward rules by binding variables in the body, which will hold for the backward rules.

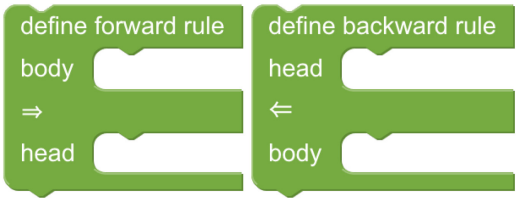

Fig. 3. Blocks to define forward and backward chaining rules.

4 Use Cases and Functionality

We briefly introduce five research-oriented scenarios that rely on mobile apps for data collection, analysis, and interpretation. Following the use cases, we categorize the distinct features of these apps, and show how the Punya components in Sect. 3 were used to address their functional requirements. A more detailed description of how these apps were built and the Punya application code for these research use cases are available at http://punya.mit.edu/#use-cases.

4.1 Use Cases

1. **Sleep Apnea Diary.** Mobile patient diaries are apps used by chronic patients to record medications, symptoms, and vitals—they provide a convenient way for patients to keep track of their health data, and, when they are integrated into Electronic Health Records (EHR), communicate health data to clinicians for longitudinal follow-up. Clinical decision support (CDS) tools, added to the EHR system, can issue recommendations to help with diagnosis and treatment. In many cases, however, it would be opportune for a patient diary to directly provide decision support: for urgent patient health issues, when wireless connectivity is lacking, or when secure integration with an EHR system is non-existent. Sleep Apnea, for example, has an estimated prevalence from 3% to nearly 50% depending on age group and sex [27], with a gold-standard diagnosis involving polysomnography [32], a comprehensive test which involves monitoring heart, lung, and brain activity, breathing patterns, and blood oxygen levels. However, more simple home sleep apnea testing may be used to indicate the diagnosis in symptomatic patients [27]. An app, running locally on the user's device and outfitted with mobile decision support (Sect. 3.4), can directly analyze the user's health, based on user input and sensor data, and issue health recommendations—in this case, a possible diagnosis of sleep apnea.
2. **Diabetes Prevention and Intervention App.** Diabetes is a chronic health condition that affects approximately 10.5% of the United States population [15]. People with diabetes are typically advised to engage in several self-management behaviors to improve their health outcomes, including healthy dieting and exercise—this is known to require personalized educational material and multitudes of data on suitable foods and types of exercise. Yet,

sustained, long-term behavior change remains challenging [11]. A possible solution involves using agile development to quickly prototype mobile apps, with the direct involvement of patient end-users, to design a mobile app that is personalized to their requirements and needs. In general, interventions that are adaptive to an individual's current psychological, social, and environmental context, are arguably in a better position to address behavior change than static or 'one-size-fits-all' approaches. Such a bespoke, context-sensitive, and data-intensive approach, requires a mobile development platform that facilitates end-user development, including patients and researchers, and allows easy integration with context-gathering sensors (Sect. 3.2) and online data sources (Sect. 3.3), such as the FoodKG [21], Ontology of Physical Activity [25], the Diabetes Mellitus Treatment Ontology [17], and large scale biomedical data repositories such as Bio2RDF [6].

3. **Remote Monitoring for Healthy Aging.** The elderly ($\geq$ 65 years old) population is expected to grow from about 725 million to approximately 1.8 billion in the next 40 years, with prevalence jumping from nearly 10% to 18% [53]. Healthy, active aging is a global challenge for the next decades from healthcare, technological and social standpoints. Many research projects have been studying *Ambient-Assisted Living* (AAL) to support healthy aging through automated activity recognition, psychophysical well-being monitoring, and injury prevention [36]. AAL relies on an array of information and communication technologies, deployed in "smart" environments outfitted with camera systems and mobile and wearable sensors, to assess the subject's condition, activate timely assistance requests to caregivers, and provide feedback and support. In AAL, smartphones often play the role of cluster heads, reading data from a variety of sensors embedded in the smartphone, such as accelerometers, gyroscopes, and orientation sensors; wearable sensors setup in typical Body Area Network (BAN) configurations [20]; and devices deployed in the room, communicating through short-range wireless links (e.g., using BLE, Sect. 3.2). In this paradigm, mobile processing capabilities are exploited to collect, integrate and enrich collected data, and run lightweight data preprocessing and mining procedures. Additionally, the mobile phone can act as a gateway towards a back-end infrastructure where larger volumes of information can be stored and analyzed (e.g., using SWoT protocols like LDP-CoAP, Sect. 3.2).
4. **Provenance of Sensor Networks.** Capturing provenance metadata in the field during deployment, calibration, maintenance, and removal of sensor platforms, is vital to helping scientists analyze and understand sensor data back. Kinkead et al. [26] discuss a research tool, in the form of a mobile app, to collect metadata about which sensor platforms were deployed where and when, as part of a larger inter-institutional effort to study water and ecosystem quality around Lake George, NY, USA. Rather than using traditional paper-based metadata collection, the mobile app facilitated real-time gathering and communication of metadata in an efficient, error-free, and standardized way: using the device's camera to identify the sensor using their QR code, and GPS to automatically identify the sensor's location. In doing so, the app allowed

accurate downstream statistical analysis based on properly calibrated and positioned sensors. The quick prototyping of such research apps by non-IT professionals (in this case, ecologists) requires an end-user development platform with a minimal learning curve. Using Punya,[6] the ecologists were able to develop the app in a matter of weeks [26]—moreover, the researchers believed that the platform could be similarly utilized to rapidly prototype observation-based mobile apps in other fields.

5. **Experience Sampling Methodologies.** Experience sampling methodologies (ESM) are utilized to build a picture of user behavior over time, relying on automated sensor input and manual user entry. For instance, Shih [50] developed a mobile app featuring ESM to study the contextual factors that influence people's privacy preferences regarding mobile apps: such as frequently visited places, specific time slots, who is around, and activities people are engaged in. As another example, Ecological Momentary Assessments (EMA) are a type of ESM that is essential to perform reliable, psychology- and healthcare-related assessments in the patient's natural environment, as they minimize recall bias and maximize the real-world validity of observations [49]—with the current ubiquity of smartphones, mobile apps have become excellent tools to perform EMA [34]. To build a complete and accurate picture of user behavior, developing ESM apps require integrating data from many different sensors and apps. Moreover, an advanced use case of ESM involves collecting detailed information on the user's social interactions, and social networks may be leveraged to automatically collect this type of information (Sect. 3.3).

4.2 Common Features

Each of these use cases brings its own requirements. We observe that, in many cases, these functional requirements are shared across apps. We briefly discuss each of these requirements, and outline how researchers can realize them within a mobile app using Punya.

– **Reading and Writing LD.** Mobile apps addressing each of the use cases above leverage Punya's ability to read and write LD (Sect. 3.3). In all cases, this data may be stored locally on the device, using a `LinkedData` model (Sect. 3.1), to support down-stream local decision-making (use case 1), realize context-sensitivity (use cases 2 and 5), perform pre-processing and data mining (use case 3), or merely for offline storage (use case 4). Subsequently, the data can be uploaded to a remote data store (e.g., EHR), when connectivity is available, to allow for remote storage, processing, and decision-making (e.g., [56]). In [26], provenance data related to sensor platform deployments are captured on-site, and then uploaded to a remote data store when connec-

[6] The authors mention App Inventor, but several Punya LD components were used.

tivity is restored. Moreover, in order to aid patients in behavior change, such as healthy dieting, use case 2 relies on reading online, large-scale LD from online data sources such as the FoodKG [21], Ontology of Physical Activity [25], the Diabetes Mellitus Treatment Ontology [17], and Bio2RDF [6].

- **Reasoning over LD.** Use cases 1 and 2 utilize the Punya reasoning components (Sect. 3.4) to write `Ruleset` blocks and apply the `Reasoner` to analyze self-reported health data, issue health recommendations, and implement context sensitivity. Use case 3 can apply either simple data mining algorithms or rule-based reasoning to perform a local pre-processing of data, required to enable edge computing scenarios and reduce both response times and bandwidth usage while increasing privacy compared to traditional centralized approaches. Moreover, in use case 4, one could also utilize the `Reasoner` to validate metadata when it is collected, e.g., to ensure that a sensor platform is still in an undeployed state at the time the app is deployed [26].
- **Integrating Sensor Data.** All use cases make use of Punya's wide array of sensor components (e.g., `BarcodeScanner`, `Pedometer`, `LocationSensor`) to achieve their goals. Some examples include: collecting health-related sensor data to aid in diagnosis (use case 1), the psychological, social, and environmental context for effective behavior change (use case 2) and accurate experience sampling (use case 5), embedded and peripheral sensor data for AAL (use case 3); and to aid in collecting sensor provenance metadata (use case 4). Data can also be annotated according to widespread modeling approaches, per the LDP guidelines, and shared via WoT protocols (use case 3). By exploiting LDP-CoAP, external data sources can be discovered at run-time and accessed, and annotated data generated by the app can be published as LDP Resources organized in containers to aggregate them, e.g., by type (OWL class), provenance (user, area, etc.), or time slot.
- **Accessing non-RDF Graph Data.** Use case 5 presents a scenario where information about study participants' social graphs and activities are relevant to privacy preferences [50] and ecological momentary assessments [49]. Understanding social graphs can be done using the `GraphQL` component to query a social network with appropriate authorization. Combined with some environmental sensing, one could imagine an ESM app that asks questions about a social relationship based on proximity to friends and family. Likewise, vocabularies like Friend Of a Friend (FOAF) [12] could also be used, assuming that participants publish their own FOAF profiles or use another platform that does so on their behalf.
- **Tracking Provenance.** Every use case could benefit from gathering provenance at the point of data collection: sensor data provided by the mobile app platform, such as time (`Clock`), location (`LocationSensor`), and ownership of the device, all can supply provenance metadata. For example, if the grand

vision that one day EHR encompass in-situ patient data on a fine-grained time scale (e.g., to the second or minute), then healthcare professionals will want to know how the data were collected, by whom, and how trustworthy those data are based on their sources. In use case 4, for example, provenance about the sensors (e.g., manufacturer, date of manufacture, model numbers, calibration data) may all prove valuable in the long run, and this information can be captured with Punya and serialized to RDF for future use.

5 Learning Materials

Punya, hosted at http://punya.appinventor.mit.edu, includes a suite of built-in tutorials for learning how to build Linked Data-aware mobile apps inspired by the use cases presented in Sect. 4. Some of these new built-in instructional materials are based on a tutorial session previously presented at ISWC 2020 [41]. For example, the RdfNotepad[7] tutorial teaches the basics of using the `LinkedDataForm` to construct and edit RDF graphs and reading and writing of RDF using the `LinkedData` component. Those interested in rule-based expert systems can explore the Sleep Apnea tutorial[8] discussed in Sect. 1. Integration of mobile app sensors and the Semantic Web of Things can be explored as part of a tutorial on LDP-CoAP.[9]

6 Usage and Community Engagement

Over 5,000 app creators have used the Punya platform since its debut in 2014. However, because the framework by design does not embed any tracking information within apps, we are unaware of whether the resulting apps have been published in mobile app stores or otherwise widely distributed. Punya has been presented as a tutorial at two meetings of the International Semantic Web Conference, once in 2015 at Lehigh University and once virtually in 2020. In the year ending March 31, 2021, there were 529 active users, of which 31 made use of the Linked Data features exclusive to Punya (Fig. 4).

Because Punya is built on MIT App Inventor, developers can tap into a large community of app developers worldwide (1,967 active community members in the 30 days ending April 1, 2021, and almost 900,000 yearly active users).[10] There is also a rich community of extension developers around MIT App Inventor, with over 3,000 extensions published [39]. Creators of Linked Data apps using Punya can leverage these extensions to further enrich their applications. Some popular extensions include the BluetoothLE extension for IoT connectivity and machine learning extensions using Tensorflow.js.

[7] http://punya.appinventor.mit.edu/?repo=RdfNotepad.
[8] http://punya.appinventor.mit.edu/?repo=SleepApnea.
[9] http://punya.appinventor.mit.edu/?repo=LdpCoapTutorial.
[10] https://community.appinventor.mit.edu.

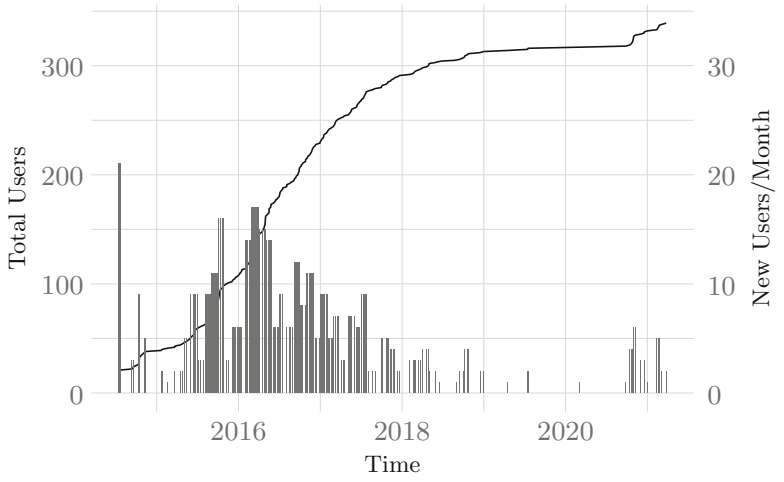

Fig. 4. Growth of users leveraging Linked Data components in Punya

7 Conclusion and Future Work

Punya is an open source[11] platform that can be used for building semantically aware research apps. A ready-to-use development environment with interactive tutorials is available online at http://punya.appinventor.mit.edu. For those interested in offline use or use without a Google account, it is possible to build and host a local copy following the instructions in the GitHub repository. Punya provides a rich suite of components for researchers to develop data-collection and expert-system type applications. Its integration of novel technologies like LDP-CoAP opens new possibilities for integrating with the Semantic Web of Things. App developers using Punya can also tap into a worldwide network of others building mobile apps via the MIT App Inventor community. We have provided a suite of tutorials to get researchers and Linked Data practitioners up to speed and building apps quickly. All the resources related to Punya, including use cases, sample apps, and tutorials, can be accessed at http://punya.mit.edu.

A continuous avenue of future work involves the integration of Semantic Web technology to further facilitate research app development—*e.g.*, we currently target an ontology-based layer on top of the `BluetoothLE` components to abstract from the relatively low-level BLE protocol. Recently, MIT App Inventor released a version that supports Apple's iOS operating system. A future iteration of Punya will build on this new platform to allow researchers to quickly prototype *cross-platform* Linked Data and Semantic Web apps. We are also looking to make the platform smarter with respect to energy consumption, aiming to move computation to either the device or server based on smart predictions [38]. We are actively building more example apps and establishing future partnerships to leverage the ease of app development with Punya to developing countries.

[11] Apache Licensed, see https://github.com/mit-dig/punya.

References

1. Adida, B., Birbeck, M., McCarron, S., Herman, I.: RDFa Core 1.1 - Third Edition). W3C Recommendation, World Wide Web Consortium, March 2015. https://www.w3.org/TR/rdfa-core/
2. Apple Inc.: CareKit. https://developer.apple.com/carekit
3. Apple Inc.: HealthKit. https://developer.apple.com/health-fitness/
4. Apple Inc.: ResearchKit. http://researchkit.org/
5. Arndt, D., Van Woensel, W.: Notation 3 (N3) Community Group (2018). https://www.w3.org/community/n3-dev/
6. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. J. Biomed. Inform. **41**(5), 706–716 (2008)
7. Blackstock, M., Lea, R.: Toward a distributed data flow platform for the web of things (distributed node-red). In: Proceedings of the 5th International Workshop on Web of Things, pp. 34–39 (2014)
8. Bobed, C., Yus, R., Bobillo, F., Mena, E.: Semantic reasoning on mobile devices: do androids dream of efficient reasoners? Web Semant. Sci. Services Agents World Wide Web **35**, 167–183 (2015)
9. Bormann, C., Castellani, A.P., Shelby, Z.: CoAP: an application protocol for billions of tiny internet nodes. IEEE Internet Comput. **16**(2), 62–67 (2012)
10. Bottoni, P., Ceriani, M.: Sparql playground: a block programming tool to experiment with sparql. In: VOILA@ ISWC, p. 103 (2015)
11. Bouton, M.E.: Why behavior change is difficult to sustain. Prev. Med. **68**, 29–36 (2014)
12. Brickley, D., Miller, L.: FOAF vocabulary specification 0.91 (2007)
13. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Proceedings of 13th International World Wide Web Conference Papers & Posters, pp. 74–83 (2004)
14. Cen, L., Patton, E.W.: Block affordances for graphql in mit app inventor. CoolThink@ JC, p. 147 (2019)
15. Centers for Disease Control and Prevention, U.S. Department of Health and Human Services, National Diabetes Statistics Report, Atlanta (2020)
16. Dominguez Veiga, J.J., Ward, T.: Data collection requirements for mobile connected health: an end user development approach. In: Proceedings of the 1st International Workshop on Mobile Development, pp. 23–30 (2016)
17. El-Sappagh, S., Kwak, D., Ali, F., Kwak, K.S.: DMTO: a realistic ontology for standard diabetes mellitus treatment. J. Biomed. Semant. **9**(1), 1–30 (2018)
18. Facebook Inc.: GraphQL: a data query language. https://engineering.fb.com/2015/09/14/core-data/graphql-a-data-query-language
19. Hartig, O., Pérez, J.: Semantics and complexity of GraphQL. In: Proceedings of the 2018 World Wide Web Conference, pp. 1155–1164 (2018)
20. Hasan, K., Biswas, K., Ahmed, K., Nafi, N.S., Islam, M.S.: A comprehensive review of wireless body area network. J. Netw. Comput. Appl. **143**, 178–198 (2019)
21. Haussmann, S., et al.: FoodKG: a semantics-driven knowledge graph for food recommendation. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 146–162. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_10
22. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al.: SWRL: a semantic web rule language combining OWL and RuleML. W3C Member Submission **21**(79), 1–31 (2004)

23. InVision Inc.: InVision. https://www.invisionapp.com
24. Kazakov, Y., Klinov, P.: Experimenting with ELK reasoner on android. In: Proceedings of 2nd International Workshop on OWL Reasoner Evaluation, pp. 68–74 (2013)
25. Kim, H., Mentzer, J., Taira, R.: Developing a physical activity ontology to support the interoperability of physical activity data. J. Med. Internet Res. **21**(4), e12776 (2019)
26. Kinkead, L., Pinheiro, P., McGuinness, D.L.: Automating the collection of semantic sensor network metadata in the field with mobile applications. In: Proceedings of 1st International Workshop on Mobile Deployment of Semantic Technologies, pp. 32–43 (2015)
27. Laratta, C.R., Ayas, N.T., Povitz, M., Pendharkar, S.R.: Diagnosis and treatment of obstructive sleep apnea in adults. CMAJ **189**(48), E1481–E1488 (2017)
28. Li, W., Seneviratne, O., Patton, E.W., Kagal, L.: A semantic platform for developing data-intensive mobile apps. In: Proceedings of 13th International Conference on Semantic Computing (ICSC), pp. 71–78. IEEE (2019)
29. Li, W.J.: Helping the helpers: a toolkit for mobile humanitarian assistance apps. Master's thesis, Massachusetts Institute of Technology (2016)
30. Loseto, G., Ieva, S., Gramegna, F., Ruta, M., Scioscia, F., Di Sciascio, E.: Linked data (in low-resource) platforms: a mapping for constrained application protocol. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 131–139. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_14
31. Marvel Inc.: Marvel. https://marvelapp.com
32. Mayo Clinic: Polysomnography (sleep study). https://www.mayoclinic.org/tests-procedures/polysomnography/about/pac-20394877
33. McGuinness, D.L., Van Harmelen, F., et al.: OWL web ontology language overview. W3C Recommendation **10**(2) (2004)
34. Miralles, I., et al.: Smartphone apps for the treatment of mental disorders: systematic review. JMIR Mhealth Uhealth **8**(4), e14897 (2020)
35. MockPlus Inc.: MockPlus. https://www.mockplus.com
36. Nilsson, M.Y., Andersson, S., Magnusson, L., Hanson, E.: Ambient assisted living technology-mediated interventions for older people and their informal carers in the context of healthy ageing: a scoping review. Health Sci. Rep. **4**(1), e225 (2021)
37. Node-RED community: Node-RED: Low-code programming for event-driven applications. https://nodered.org
38. Patton, E.W.: Energy aware reasoning agents for the mobile semantic web. Ph.D. thesis, RPI (2016)
39. Patton, E.W.: A look at component usage in MIT App Inventor (2020). http://appinventor.mit.edu/blogs/evan/2020/12/20/component-usage-mit-app-inventor. Accessed 01 Apr 2021
40. Patton, E.W., McGuinness, D.L.: A power consumption benchmark for reasoners on mobile devices. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 409–424. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_26
41. Patton, E.W., Scioscia, F., Van Woensel, W.: Building mobile semantic web apps with Punya. In: Proceedings of ISWC 2020 Tutorials (2020)
42. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. (TODS) **34**(3), 1–45 (2009)
43. Praino, E., et al.: SScEntry: a personal disease diary app for Systemic Sclerosis patients. Ann. Rheum. Dis. **79**, 558–559 (2020). eULAR 2020 European eCongress of Rheumatology

44. Proto.io Inc.: Proto.io: Prototyping for all. https://proto.io
45. Qualtrics Inc.: Qualtrics: XM OS - Experience Design and Improvement. https://www.qualtrics.com
46. Ruta, M., Scioscia, F., Di Sciascio, E.: Enabling the semantic web of things: framework and architecture. In: 2012 IEEE Sixth International Conference on Semantic Computing, pp. 345–347. IEEE (2012)
47. Sambra, A., et al.: Solid: a platform for decentralized social applications based on linked data. Technical report, MIT CSAIL & Qatar Computing Research Institute (2016)
48. Shelby, Z.: Constrained RESTful Environments (CoRE) Link Format. RFC 6690, Internet Engineering Task Force, August 2012
49. Shiffman, S., Stone, A.A., Hufford, M.R.: Ecological momentary assessment. Annu. Rev. Clin. Psychol. **4**, 1–32 (2008)
50. Shih, F.: Exploring mobile privacy in context. Ph.D. thesis, MIT (2015)
51. Sittón-Candanedo, I., Alonso, R.S., Corchado, J.M., Rodríguez-González, S., Casado-Vara, R.: A review of edge computing reference architectures and a new global edge proposal. Futur. Gener. Comput. Syst. **99**, 278–294 (2019)
52. Steve Speicher and John Arwe and Ashok Malhotra: Linked Data Platform 1.0. https://www.w3.org/TR/ldp
53. United Nations Department of Economic and Social Affairs: World Population Prospects 2019. https://population.un.org/wpp/
54. Van Woensel, W., Abidi, S.: Benchmarking semantic reasoning on mobile platforms: towards optimization using OWL2 RL. Semantic Web **10**(4), 637–663 (2019)
55. Van Woensel, W., Roy, P., Abidi, S., Abidi, S.: A mobile and intelligent patient diary for chronic disease self-management. In: Studies in Health Technology and Informatics, vol. 216 (2015)
56. Wilkinson, M., Vandervalk, B., McCarthy, L.: The semantic automated discovery and integration (SADI) web service design-pattern, API and reference implementation. Nat. Preced. 1 (2011)
57. Wolber, D., Abelson, H., Friedman, M.: Democratizing computing with app inventor. GetMobile: Mob. Comput. Commun. **18**(4), 53–58 (2015)
58. Yus, R., Bobed, C., Esteban, G., Bobillo, F., Mena, E.: Android goes semantic: DL reasoners on smartphones. In: Proceedings of 2nd International Workshop on OWL Reasoner Evaluation, pp. 46–52 (2013)