decode

Final DECODE app release. App published on multiple platforms

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

D4.14 Final DECODE app release. App published on multiple platforms

Version Number: V1.0

Lead beneficiary: DRIBIA Data Research S.L.

Due Date: October 31st 2019

Author(s): Oleguer Sagarra, Xavier Hoffmann, Xavier Clotet, Pol Colomer (DRIBIA)

Editors and reviewers: Javier Rodríguez, Pau Balcells (IMI), Jim Barrit (Thoughtworks), Pablo Aragón, Rohit Kumar (Eurecat).

Dissemination level:		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: Francesca Bria (DECODE Project Coordinator)
Date: 29/10/2019

This report is currently awaiting approval from the EC and cannot be not considered to be a final version.

Table of Contents

Introduction

This short document is intended to be distributed with the DECODE APP v2.0 and its attached documentation (see the repository and website). The app is distributed in both Google Play and Apple Store and is available for the general public in European countries.

The app is a meeting point of several branches of work present in the DECODE project Work Packages (WPs). It integrates the technological tools developed in WP3 (*Blockchain for decentralised data and digital identity management*) and WP4 (*DECODE IoT node distributed hardware & software platform*) and it is intended to serve (or be extendable to do it) the pilots designed in WP5 (*Pilots and Participatory Innovation*) following the recommendation enunciated in WP2 (*Decentralised Governance and Economic framework: Commons data platforms for digital sovereignty*) and the design principles worked throughout WP1 (*Privacy aware citizen centric distributed architecture*).

The app has been designed incorporating the conclusions of user research (see deliverable D4.10[1]) and with the aim to be extendable, modular and flexible enough so all the cases researched and developed to exemplify the DECODE technology can be fitted into it (see deliverables D5.5[2] and D5.6[3]). Not only that, but it is intended to be extendable to other cases thanks to its open source license and modular architecture.

Currently, two services are fully implemented in the app corresponding to real world pilots tested in Barcelona: BCNNow[4] (lead by Eurecat) and DDDC[5] (lead by UOC). The full design for integration for the Citizen Science Data Governance IoT pilot[6] (IoT pilot for short, lead by Thingful), the third pilot, is sketched in this document, as an example of how any interested developer should adapt the current codebase to needs posed by a new project.

This document is structured as follows: The first part covers the general implementation of the taxonomy existing in the DECODE pilots, a general schema of its parts and the actual detailed implementation of the schema for the two services currently implemented in the app. Then, a second part follows where the technology used within the app is presented as well as the methodology followed. Lastly, the document is closed with an example based on the IoT use-case on how the app should be

[1] See DECODE Website for deliverables. In particular https://www.decodeproject.eu/publications/uxui-decode-app-development-integrated-bcnnow

[2] https://www.decodeproject.eu/publications/deployment-pilots-amsterdam

[3] https://www.decodeproject.eu/publications/deployments-pilots-barcelona

[4] http://bcnnow.decodeproject.eu

[5] http://dddc.decodeproject.eu

[6] http://iot.decodeproject.eu

extended to add additional features. From there, a future roadmap is laid out, including the possibility of other services such as the pilots developed in Amsterdam to be integrated in the system.

Part 1: General framework

General purpose of the app

The DECODE app is intended to be a multilingual piece of software for users to easily use DECODE functionalities. It represents the merging point of the diverses DECODE researched technologies, such as cryptographic languages (Zenroom), schemes (COCONUT), credential management APIs as well as distributed enhanced services (such as petitions API and IoT encoder, IoT policy store and IoT distributed store). It is built based on four core principles.

Minimization: The main purpose of the app is the easy and intuitive management of attributes (*aka personal data),* credentials (*aka proofs over attributes*) and external apps needing them. As such, any other added service not involving attributes or credentials shall be leveraged externally to the app via appropriate back-ends. Those, however, must adapt to a common standard to ensure generalization.

Self-containment: Also, in order to preserve users' privacy, the app does not rely on any back-end service to work. However, it uses the back-end services of the use cases it is used for.

Context: The app is built on the principle that *users activate the app to interact with known services*, following a user journey based on the paradigm *service-app-service*. For more details see the details of the UX design in deliverable (D4.10 *UX/UI for DECODE app development integrated to BCNNow*).

Customization: The app implements the requirements for usability for two concrete use cases, BCNNow and DDDC in three different languages (Catalan, Spanish and English). Yet, it is designed with the potential to be generalized to other use cases. A fully fledged example on how to do so is included: the IoT case.

In order to achieve this, we present, in the following section, a general taxonomy of the elements involved in the app together with its practical implementation to the two considered use cases. Any service wishing to integrate the DECODE technology, shall need to perform the same mapping here present for its concrete use case (an example for the IoT case is provided in the final part of this document).

App taxonomy

The DECODE ecosystem can be understood as roughly comprised of 6 main elements, derived from the ongoing research (see D1.4[7] *First version of DECODE architecture* and D1.5[8] *Intermediate version of DECODE architecture*):

- **Services/apps:** External services that wish to use DECODE technology and integrate on the ecosystem. Those services can in turn have supporting *back-end* services that help them provide the purpose they are built for.
- **Attributes:** Any kind of data related to a user in the DECODE universe. Those attributes must be normalized according to a common set of standards, which we call the Atlas (see below). Also, those attributes can be *transformed* into *derived attributes*, mostly for the purpose of aggregating information. For example, an age or an age range can be transformed from a birth date, the same as a city can be derived from a precise location. Also, attributes can be included into credentials and be authorized (see more below).
- **Credentials, Issuers, Verifiers and contracts**: Credentials are statements proposed by users. Those can be signed and hence endorsed by an Issuer and can be verified by a Verifier (entity that provides proof of the integrity and validity of the statement, not on its contents). All this is managed through cryptographic contracts, which can in turn be inspected thanks to Zencode natural language syntax (see more on deliverable D3.6[9] *Smart Rules Implementation Evaluation of Prototypes and Integration*).

The properties that those elements can have are sketched below in figure 1.

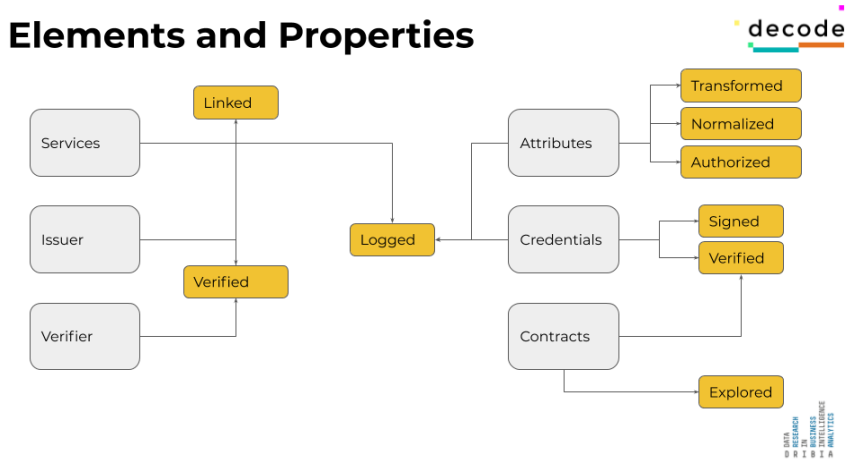

Figure 1: DECODE taxonomy elements and properties

[7] https://www.decodeproject.eu/publications/decode-architecture-first-version
[8] https://www.decodeproject.eu/publications/intermediate-version-decode-architecture
[9] https://www.decodeproject.eu/publications/smart-rules-implementation-evaluation-prototypes-and-integration

Relations

Any service or app wishing to integrate itself in the DECODE ecosystem, must fit into the schematic relationship scheme provided below in figure 2, and explained here.

The starting point is always the user, which can input some attributes into the DECODE app (potentially personal data). Those attributes must be normalized to conform to a common specification, called Atlas. The Atlas is a reference (currently implemented in the form of a json file) that contains the basic information relating services to attributes and credentials (see more about it below). Also, those base attributes can be transformed into less informative ones (to preserve a certain degree of anonymity) in order to be shared with services. For instance, transforming a given address into a city of residence.

A user can provide certain attributes (a user name, a password, a code, an identity card) that must be authorized by a recognized Issuer (for instance a city council) to emit a signed credential. Those credentials can be verified by anyone wishing to validate their integrity via a Verifier instance. Once a credential is signed, it can be used to provide proof of statements such as "I am over 18 years old". This statement, uses a derived attribute from a birth date, and is signed by an Issuer. All this process is governed via cryptographic contracts, written in Zenroom language.

Once a user has credentials, it can use those credentials to make use of the federated services/apps, which can optionally rely on supporting services (all outside of the app). As a service is added to the DECODE ecosystem, it can declare its link to other existing services, so users can benefit from both (for instance, a new service to visualize data from using another declared federated application).

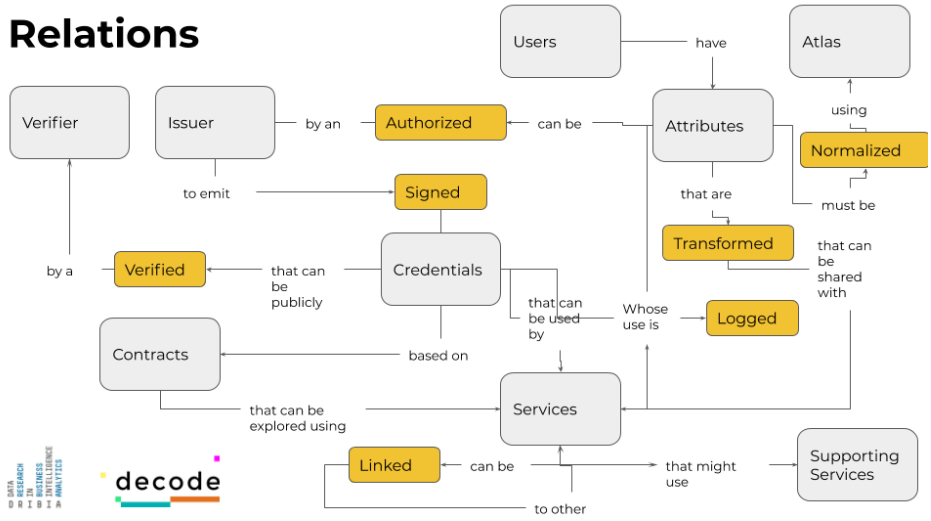

Figure 2: DECODE relations among elements

The diagrams above might look a bit abstract, but they are an essential part of the system. Identifying and placing different elements into them makes the adaptation of any given service to the DECODE ecosystem much easier. The strength of the above schema relies at the same time on its simplicity and its generality. On the one hand, it allows non expert users to rapidly identify the needed pieces and relations to take into account when adapting a service to DECODE. On the other, the relation between services, attributes and credentials is broad enough so a variety of real world scenarios can be fitted into it. In the following, we describe the DDDC and BCNNow pilot adaptations to the above proposed schema to exemplify the aforementioned strengths and give a bit of practical explanations about the taxonomy.

Pilot mapping: DDDC & BCNNow

The above relations must be mapped to actual existing instances in the real world for any service or app wishing to integrate itself into the DECODE ecosystem. Below, we provide explicit mappings of those instances for the BCNNow and DDDC use cases, belonging to the pilots developed in Barcelona.

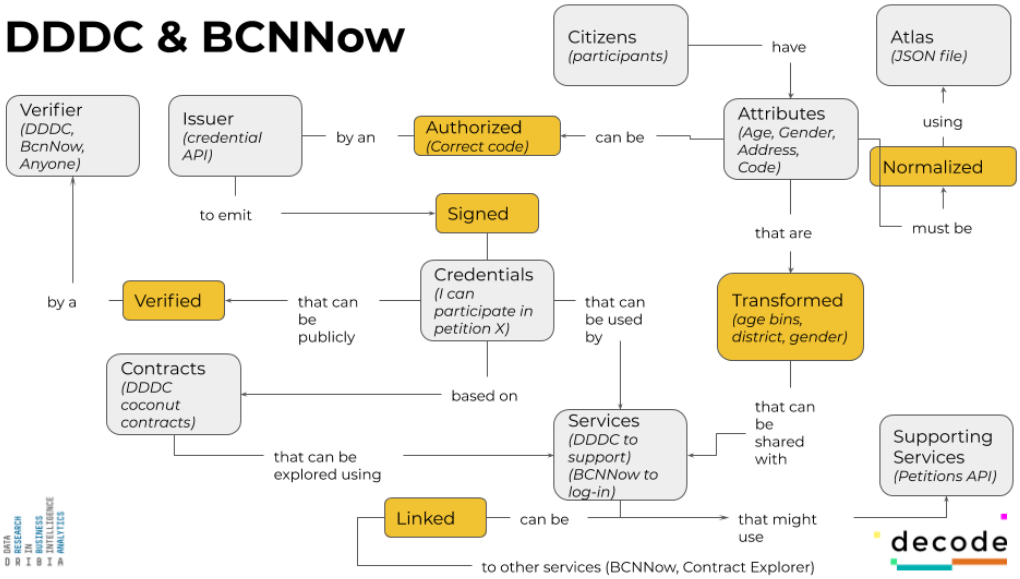

Figure 3: DECODE relations among elements mapped to the DDDC and BCNNow case

In these use cases, users are the citizens of Barcelona. They input their address, age and gender into the app and the app transform those values into approximate locations and age ranges (*derived attributes)*. The users participate into the DDDC deliberation process, and are given a unique participatory code to be able to support petitions displayed on the site.

This code is used to obtain a credential, signed by the DECODE credential issuer (see Technological implementation section for details below) that validates the statement "*I can participate to support petition X*", being X a petition chosen by the user. The credential, can be presented to any instance wishing to verify it, which can be done at a public endpoint of the credential issuer itself (currently implemented as http://credentials.decodeproject.eu). At the moment of asking for a credential, the user is given the option to share derived data (gender, blurred address and age range) with the credential issuer for statistical purposes, hence effectively building a data commons[10] for the improvement of the system.

Once the user is in possession of the credential, it can use it for two actions in two different apps or services (which are linked):

1. It can support a petition chosen from the DDDC website. To do so, DDDC uses the supporting service that manages the petitions support in a distributed way over a ledger (see deliverable D3.10 *Implementation of Blockchain platform and ABC in DECODE Pilots*).

2. It can use the same credential to log into BCNNow website. Upon doing so, the site also asks for some derived data that the user can share in order to personalize her experience.

All those actions can be done via the app, either by clicking on a link with the appropriate *handle* while being on a mobile environment *(decodeapp://…)*, or by scanning a QR code that the app can recognize. Below, we provide the technological implementation details that explain the entire process.

The syntax for QR compatible applications is specified in the technical documentation of the app, which is hosted on its public repository.

[10] By *Data Commons* we refer to an aggregated dataset of donated information by users, which (a) respects their individual privacy while at the same time (b) allows its exploitation for public/common good objectives such as optimizing a platform to their user base or detecting excluded profiles and understanding better the city by mapping urban problems among others.

Part 2: Technological implementation

Development methodology

The app has been developed using *agile methodology* in concrete, a Scrum flavor. The development has been organized into sprint of 10 able days each, with allocation of points per task using a Fibonacci scheme.

For each sprint, 70 points were allocated and executed. In parallel, UX design and user testing sessions were carried out and its conclusions used to inform design and development of the app. For each sprint, regular meet-ups of the team were done and a final presentation of results on day 10 was carried out with the presence of the Tech leader of the project, Dyne foundation. This process was combined with the regular tech stand-up meets every week with other project partners, to ensure integration of all involved components on which the app relies.

Tech stack

The DECODE APP is a piece of software developed from scratch, in React Native, without dependencies on third party servers and using a "stateless, pure functional component" paradigm, where all application state is centralized in the Redux store.

The DECODE depends on the following technologies:

- Redux for state management
- Reselect for cacheable (memoized) access to the state
- Redux Thunk for asynchronous action dispatching
- Redux Persist for saving state to permanent storage
- AsyncStorage from React Native Community for native key-value permanent storage
- Plain Fetch API for communication with services
- React Navigation for navigation between screens and menus
- React Native Vector Icons as icon library set
- Styled Components for styling and theming
- Ramda for utilities
- moment for date & time handling
- react-i18next for multilanguage
- react-native-sentry for sending crash logs (requires a Sentry server)
- react-native-splash-screen to maintain the splash screen while JS is loading
- react-native-onboarding-swiper for the app intro caroussel
- react-native-walkthrough-tooltip for the screen tooltips walkthrough

- react-native-camera for QR scanning
- react-native-date-picker as cross platform component for date selection
- react-native-render-html to support HTML coming from application description fields
- react-native-loading-spinner-overlay as loading indicator
- react-native-keyboard-aware-scroll-view to fix layout issues when a KeyboardAwareView has a ScrollView inside

Additionally, there are other development dependencies:

- yarn for dependency management
- babel for code transpiling
- eslint and prettier for code style
- jest for unit testing
- fastlane for store deployment automation
- sentry for crash reporting

Finally, the app depends on Zenroom for all cryptographic operations, via its handlers, which is the language specifically developed for the DECODE project by our partner Dyne.
Please note that further technical details of the app can be found on the repository documents.

Below, we provide a general reference of all the technological elements involved in the implementation of the DECODE taxonomy.

App

Conceptually, the components of the system are simple to understand: An app, different sources of data, a credential service and a set of external services.

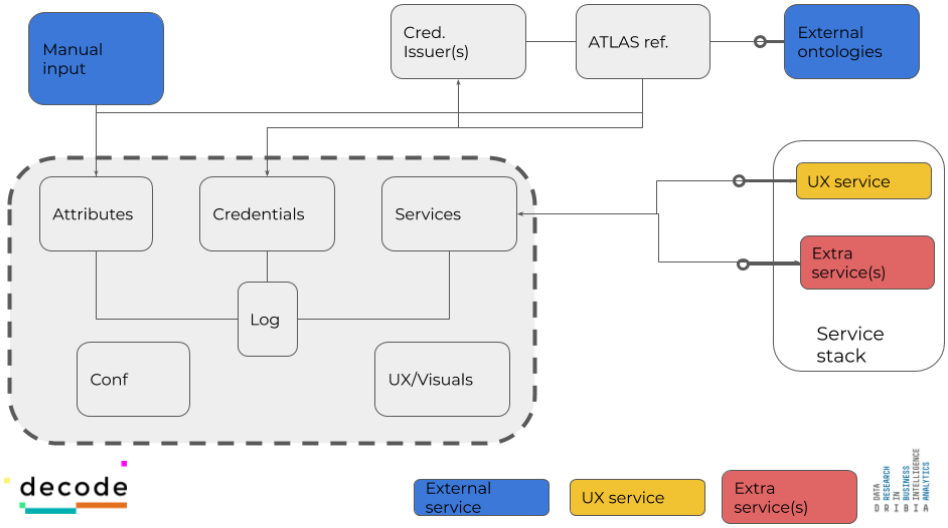

Figure 4: DECODE App basic elements

Sources of data

Those can be either manual input by user (current implementation), or imported from other services. The key point is that the data used in the DECODE ecosystem need to respond to a common Atlas (see below).

Credential service

This service is already implemented and dockerized here, and has a domain (credentials.decodeproject.eu). You can find its documentation here. It issues certificates with given credentials interacting with the App. It also stores information relating to the users that asked for a credential for statistical purposes in an aggregated way.

Atlas

This is the service where all the conceptual objects of the DECODE universe exist which maps to the app taxonomy. It has the form of an internal JSON that could later be externalized in the form of an online API endpoint. The key characteristic of the Atlas is:

- No synonyms can exist in it (each object cannot have two names).
- It can be extended by adding new elements that do not exist in it, but its original elements cannot be modified without the agreement of all the parties of the project.

The use of the Atlas is capital as it defines the type of attributes (data) that DECODE can handle. This in turn determines the credential logic, the zenroom contracts and also the configuration of the external services.

The structure of the Atlas is self-explanatory from the JSON itself, but essentially contains three elements:

- Attributes declaration: All types of attributes that the app can understand are declared there, they can be of various types (float, enum, integer, string, bins) and can be either original or derived.
- Service declaration: All services (call applications in the front-end) are declared here, together with their reference websites, APIs as well as attributes that the service uses in the context of the DECODE project.
- Translations: Finally, all translations specific to the attributes and services are also included in the doc.

The Atlas is currently implemented as a plain JSON, but could be replaced by a fully fledged ontologies API external service/reference. Additional details on the ATLAS can be found in the technical documentation.

Service stack

Roughly, any service is composed of three elements:

- A visualization service.
- A website that explains and manages the service. All non strictly personal steps and actions can run there.
- Any microservices to support the service.

Below, we provide the mapping used for the DDDC & BCNNow use cases.

DDDC website

In our case we use DDDC, which is a DECIDIM instance. The website is up and running. It is written in Ruby and fully dockerized (see repository). To set-up a petition, it interacts with the APP, as well as to the credential and petition services. All via documented OpenAPI-dependent specs. The website has a Graphql API endpoint that accepts queries.

Petition service

For the DDDC case we use the petitions module API (petitions.decodeproject.eu). The service is owned by Dyne, it is hosted up and running. It provides interface for secure petition handling, backed on a distributed ledger system.

BCNNow

BCNNow (bcnnow.decodeproject.eu and repository) is used by the Barcelona pilots to visualize data. It is owned by Eurecat and currently running. It allows guest log-in (no credentials needed), as well as personalized views with appropriate credentials for both the IoT and DDDC case. Upon log-in, it also allows users to share data to obtain personalized dashboard views.

Part 3: Future roadmap

In the user sessions carried to flesh out the app structure, a recurring demand by users was the adaptation to the DECODE ecosystem to more apps and services. In this closing section, we thus add all the necessary indications on how to do so, focusing on the remaining use case in Barcelona, the IoT pilot.

This example might be used to perform a tutorial session on the DECODE Tech Symposium, related to the upcoming deliverable D5.8 *DECODE Developers Conference: Opening up the DECODE App and tools to third party developers and entrepreneurs*. It will also be featured on the App website and repository. The example and the mentioned sessions will be used to engage external developers into adapting services and apps into the decode ecosystem. Also, it must be noted that currently in the world there are more than 60 running instances of the DECIDIM software, of which DDDC pilot is an instance. Hence, with minimal work the results of the BCN pilots could be scaled to reach vast communities, including also the extension of BCNNow to other municipalities (being it fully general and open source).

Adding new services to the app

The general steps to add a service to the app are simple and listed below. As stated, the app has been developed with the idea that *the last mile must be run by the service wishing to be integrated*. Once a service is fully up and running, our estimate is that with good knowledge of react programming, no more than 5 days of coding work are needed for the task (excluding development of back-end services).

The general scheme to adapt a service or app to the DECODE app is as follows:

1. **Conceptual work:** The app wishing to integrate to the DECODE app ecosystem must first identify the main elements of interaction and adapt them to the DECODE taxonomy.
2. **Back-end development:** All the back-end services for the app must be made ready, with available end-points and programmatic interaction via documented APIs. At this point, client code must be written for the app to be able to consume those services.
3. **Atlas declaration:** The use case must be integrated into Atlas. Identifying the app/service name, the data used, its transformations as well as the auxiliary back-end services used. If any data transformation from basic user attributes is needed, then the functions for the app to make the transformation (the *converters*) must be written. Also, the proper zenroom contracts to use must be added to the source code in the relevant place. Lastly, the code for the back-end services' API clients must also be included in the app.
4. **Business logic:** In this step, the state of the app and its business logic must be set, which makes use of (a) the app internal attribute, credential and service management tools and (b) the earlier written API clients code developed in step 2.
5. **Screen logic:** This step declares the screens for the user to perform the actions needed within the DECODE app to integrate the service. It may use the proposed templates or additional ones. In those screens, the business logic defined and implemented in step 4 is used.
6. **Action triggers:** This step declares the actions that will trigger the DECODE app activation, either via a handle (on mobile navigation) or a QR code scan conforming to the specifications.
7. **Styling:** If wishing to change the theme, the adequate file can be adapted.
8. **Testing and deployment to stores:** Once all the integration is complete, it is important to pass the testing suite as a prior step to publication in stores.
9. **Contributing:** Once all the earlier steps are fulfilled, a PR may be submitted to the main repository so the fork can be merged into the project.

Below we provide details on the first phase of the process: The conceptual mapping of the different components. To obtain specific details about the actual code changes needed to perform the implementation, we refer the reader to the appropriate section

in the repository of the app. Two examples are provided there, a simple adaptation to a log-in for a generic website, and a more advanced one to integrate the IoT case.

Pilot mapping: IoT

The IoT service taxonomy is easy to map to the earlier defined taxonomy, and is done in the figure 5 below.

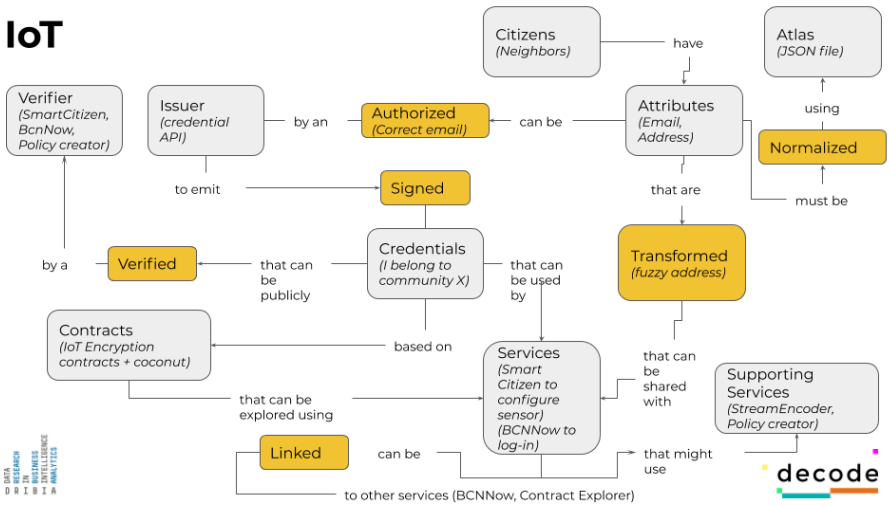

Figure 5: DECODE relations among elements mapped to the IoT and BCNNow case

Basically, the use case is as follows: Citizens want to organize themselves in communities of neighbors. Each citizen owns one or more than one sensor and wishes to generate several versions of sensor readings. One coarse grained that the general public can access (community "public") and one for the internal community (community "X"). The rules of sharing for the internal community are pre-defined, and the authorization to join a community is given via a code, which the authorized users know from external sources to the DECODE app (for example, an in-person meeting).

Hence, to start sharing data, the users need:

- A sensor configured to send data to an endpoint with an identifier.
- A code allowing them to be authorized to obtain a credential that proves they belong to a given community.
- A website where people can visualize the different versions of the datasets, depending on the credentials used.

While apparently its use case is different from the DDDC one, as we will see, most of the developed services can be reused. In particular, we will reuse the logic of log-in to

BCNNow, as well as the logic of obtaining a credential from the very same credential issuer that the DDDC pilot uses. The steps to do are detailed below:

- Obtaining a credential ("I belong to community X") map the ones to obtain a credential to participate to petitions in DDDC.
- Setting up the supporting services (StreamEncoder, Policy Store, IoTStore) those are new steps, most of which are managed by the PolicyStore micro-service.
- Linking to the visualization service website (BCNNow): Those are already implemented in the app.

Importantly, the service is already operating using a web-app interface for users, and it has been satisfactory tested with users both on the feasibility, technological and usability fronts (for technical details see D3.9[11] *IoT privacy-enhancing data sharing: integration with pilot Infrastructures*).

Back-end development

The main flow followed can be seen in detail in the technical document in the repository here. Its elements are presented below.

Smartcitizen website

The smartcitizen website is the base website of the pilot. It has a repo with the dockerized API and an API. The entry point for the pilot is this page.

Stream encoder

This is the piece of code that encodes the streams of IOT data published by the sensors. Owned by Thingful, deployed and open source.

Policy service

This is the service that basically provides policies to the encoder. It is developed by Thingful and open source.

Data store

This is the service that store the encrypted data streams. It is deployed and open source and owned by Thingful.

Other technical steps

Once reached this point where (a) the conceptual work is done and (b) the back-end services are ready, the missing step is the actual implementation. Logically, the more

[11] https://www.decodeproject.eu/publications/iot-privacy-enhancing-data-sharing-integration-pilot-infrastructures

diverse a use case is, the more work it might involve. However, the presented paradigm of the DECODE app managing credentials, attributes and services is very flexible and can accommodate a wide variety of use cases. Hence, expected development work is small. For the IoT case, the specific steps to follow can be observed in the detailed document provided in the repository of the App.

Conclusion

This document accompanies a keystone deliverable of the DECODE project, embodied in the app that allows users easy interaction with the developed project framework. This app effectively gathers and merges the team effort of many partners in the consortium, and synthesizes into a real, usable product the learnings derived from the research carried out in a variety of fronts.

The publication of the repository, the first complete version of the DECODE App as well as the accompanying website which explains the app technology for non advanced users mark an important step in the project. Milestones MS13 (*DECODE app final deployment)*, MS14 (*DECODE app website*) and MS15 (*DECODE final distributed architecture*) make the DECODE components ready to be distributed and adopted by a growing community of developers, to which it will be presented in the upcoming DECODE Tech Symposium, to be held in Torino on the 5th and 6th of November and related to deliverable D5.8 *DECODE Developers Conference: Opening up the DECODE App and tools to third party developers and entrepreneurs*.

The contributions of the different branches of the DECODE project work packages and partners have allowed to build a piece of open and free software that is preliminary validated by users, effectively respects their privacy, and is made modular enough so service can federate with it. The challenges were big, but we believe we have built an example that not only works incorporates state of the art cryptography, but also implements user demands into real world pilots.

The app does implement solutions for the pilots while preserving a general framework and philosophy allowing the work to be carried out beyond the particular cases which have been envisaged for it. In particular, other uses such as the ones designed and implemented in the Amsterdam pilots, presented in D5.5 (*Deployment of Pilots in Amsterdam*), or the final adoption by a large platform such as SmartCitizen as well as the many (60+) instances of the DECIDIM software active throughout the world provide a promising landscape for further development. Last but not least, the implementation of an IoT scheme that effectively makes possible data sharing at different granularities, thus enabling real data sovereignty, and its implementation embodied here, marks yet another important feat for the project. Such an event also opens a world of possibilities for privacy aware IoT solutions in social and industrial environments.

At this point and as we approach the end of the project, we can say that most of its ambitious objectives have been fulfilled. Now, it is up to the community to adopt and extend a service which can potentially serve many different scenarios, as exemplified with the actual implementation of the project pilots.