



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

Bachelor's Final Project

Title	The Architecture of eReuse.org and the Development of DeviceHub
Author	Xavier Bustamante Talavera
Director	Leandro Navarro
Co-director	David Franquesa
Centre	Facultat d'Informàtica de Barcelona
Department	ESSI
Date	05/2016

ABSTRACT

In this project, we design a solution to manage devices (such as desktops, laptops, and smartphones) with the main objective of ensuring reuse and final collection. It is part of eReuse.org, a program funded by the European Union with the mission of reaching the circular economy of electronics.

Manufacturing digital devices consumes a high quantity of finite resources; in addition, in a too-short period, they are thrown away instead of being repaired, refurbished, or upgraded. In the best of cases, they will be properly recycled; however, the majority will be informally recycled in the developing world or end up in landfills, jeopardising the locals and the environment. Many factors contribute to this, such as the lack of information, software, traceability solutions, and standards.

eReuse.org aims to empower and engage people around the world to create local communities, which bootstrap electronic reuse reaching circular electronics. We design the software architecture supporting eReuse.org, by enabling reuse management and creating a global traceability standard; we develop one of the architecture's components: DeviceHub, an IT Asset Management System focused on reusing.

The system is co-designed, extended, and used by many organizations in different countries, and contributes to empowering the reuse and recycle sector, reducing WEEE and promoting social change when reusing is done to social projects.

Keywords: Human-centered computing → Social content sharing; Hardware → Impact on the environment; Information systems → Open source software; Networks → Location based services; Social and professional topics → Governmental regulations;

RESUM

En aquest projecte desenvolupem una solució per gestionar dispositius (com ordinadors, portàtils o smartphones) amb l'objectiu principal d'assegurar la reutilització i la recollida final. El projecte és part d'eReuse.org, un programa finançat per la Unió Europea amb l'objectiu d'assolir l'economia circular en l'electrònica.

Manufacturar dispositius digitals consumeix una gran quantitat de recursos finits; a més, en un període de temps massa curt, aquests són tirats en comptes de ser reparats, restaurats o millorats. En el millor dels casos, seran reciclats correctament; en canvi, la majoria seran reciclats de manera informal en un país del món subdesenvolupat o acabaran en abocadors, amenaçant a la gent de la zona i al medi ambient. Hi contribueixen diversos factors, com la falta d'informació, software, solucions de traçabilitat, i estàndards.

eReuse.org vol apoderar i motivar a gent al volant del món per crear comunitats locals, capaces de potenciar la reutilització electrònica, aconseguint així l'economia circular en l'electrònica. En aquest projecte dissenyem l'arquitectura de software que dona suport a eReuse.org, permetent la gestió de la reutilització i creant un estàndard global de traçabilitat; desenvolupem també un dels components de l'arquitectura: DeviceHub, un sistema de gestió d'actius d'IT (de l'anglès IT Asset Management System) focalitzat en la reutilització.

El sistema està essent codissenyat, estès i utilitzat per diverses organitzacions en diferents països, i contribueix en millorar el sector de la reutilització i el reciclatge, reduint deixalles electròniques i promovent el canvi social quan els beneficiaris de la reutilització són projectes socials.

RESUMEN

En este proyecto desarrollamos una solución para gestionar dispositivos (como ordenadores, portátiles y smartphones) con el objetivo principal de asegurar la reutilización y la recolección final. El proyecto es parte de eReuse.org, un programa financiado por la Unión Europea con el objetivo de alcanzar la economía circular en la electrónica.

Manufacturar dispositivos digitales consume una gran cantidad de recursos finitos; además, en un periodo de tiempo demasiado corto, éstos son tirados en vez de ser reparados, restaurados o mejorados. En el mejor de los casos, serán reciclados correctamente; en cambio, la mayoría serán reciclados de manera informal en el mundo subdesarrollado o terminarán en vertederos, amenazando a los lugareños y al medio ambiente. Diversos factores contribuyen en ello, como la falta de información, software, soluciones de trazabilidad, y estándares.

eReuse.org quiere empoderar y motivar a gente alrededor del mundo para crear comunidades locales, capaces de potenciar la reutilización electrónica, logrando así la economía circular en la electrónica. En este proyecto diseñamos la arquitectura de software que da soporte a eReuse.org, permitiendo la gestión de la reutilización y creando un estándar global de trazabilidad; desarrollamos también uno de los componentes de la arquitectura: DeviceHub, un sistema de gestión de activos de IT (del inglés IT Asset Management System) focalizado en la reutilización.

El sistema está siendo codiseñado, extendido y usado en diversas organizaciones en diferentes países, y contribuye en mejorar el sector de reutilización y el reciclaje, reduciendo desperdicio electrónico y promoviendo el cambio social cuando los beneficiarios de la reutilización son proyectos sociales.

AGRADECIMIENTOS

Habría sido imposible recorrer este camino sin mis padres, cuyo amor y soporte incondicional me ha hecho crecer como persona, dándome la base que dio comienzo a mi aventura.

Ryan ha sido mi fiel compañero, quien siempre me ha ayudado y animado. Ha estado ahí cuando lo necesitaba, iluminando el camino cuando eran penumbras y tirando de mi cuando no podía más.

Querría agradecer a David Franquesa y a Leandro Navarro, los directores del proyecto, que me recogieron cuando era todavía un niño y me dieron la oportunidad de brillar. Mucho de lo que he aprendido es gracias a ellos, y estoy orgulloso de trabajar a su lado.

Nada habría sido posible sin el equipo de eReuse.org: Nataly y Timmy, Rosario, Raquel, Óscar, Cèlia, Mayra, Santiago y Adrià. En especial a Santiago, por todo lo que me ha enseñado de tecnología y por sus muchísimas contribuciones en el proyecto; y a Adrià, quien me ha demostrado que la voluntad lo es todo.

Debo agradecer a TxT, quienes me enseñaron que el poder para ayudar y mejorar está en nosotros mismos. En especial pido gracias a Fermín Sánchez, a David López, a Christian y a Alejandro.

Agradezco a la “European Community Framework Programme 7, Collective Awareness Platforms for Sustainability and Social Innovation (CAPS)” en el proyecto “Collective enhanced Environment for Social Tasks” (CHEST), por la aportación económica que nos ha permitido despegar.

Finalmente debo mencionar a los amigos de la universidad, la ya no tan famosa “chupipandi”, cuyos desvaríos, charlas y fiestas han hecho de este recorrido, un camino que ha valido la pena andar.

INDEX

Abstract	1
Resum.....	2
Resumen.....	3
Agradecimientos	4
Index.....	5
List of figures	8
List of tables	12
List of code listings	13
1 Introduction.....	15
1.1 Objectives	16
1.2 Stakeholders.....	17
1.3 Scope and requirements	17
1.4 Limitations and risks.....	18
2 State of the art	20
2.1 Definitions.....	20
2.1.1 Reuse and recycle	20
2.1.2 Device	20
2.1.3 IT Asset Management System	20
2.1.4 Inventory	21
2.2 Justification.....	21
2.2.1 The life cycle process.....	22
2.3 Existing external solutions.....	23
2.3.1 GLPI.....	23
2.3.2 Ralph.....	24
2.3.3 OCS Inventory NG	24
2.3.4 Spiceworks.....	25
2.3.5 Landesk.....	25
2.4 Technical background.....	25
2.5 Conclusion	26
3 Methodology, temporal planning, and technical competences.....	28
3.1 Methodology.....	28
3.2 Calendar	28

3.3	Organization of the project	28
3.4	Plan	29
3.5	Deviations	32
3.6	Technical competences	32
4	Budget.....	34
4.1	Human costs.....	34
4.2	Non-human costs	34
4.3	Risks.....	35
4.4	Total cost.....	35
4.5	Control management.....	36
5	eReuse.org architecture.....	37
5.1	Components	37
5.1.1	Device Diagnostic and Inventory (DDI).....	37
5.1.2	DeviceHub	38
5.1.3	Global Record of Devices (GRD).....	41
5.1.4	TransferHub	41
5.1.5	eReuse.org App.....	42
5.1.6	Directory of Collection Points (DCP).....	43
5.2	Representation of the knowledge and API.....	44
5.2.1	Requisites for the exchange of knowledge	45
5.2.2	Definition of the Ontology.....	45
5.2.3	The URI as the main identifier.....	54
5.2.4	API	56
5.3	Managing reusing.....	57
5.3.1	Preparation for the reuse	57
5.3.2	The transferal process	62
5.3.3	The collection.....	67
5.4	Ensuring reusing and recycling.....	67
5.4.1	The Hardware Identifier (HID).....	68
5.4.2	Acknowledging final usage.....	70
5.4.3	Tracing devices	70
5.4.4	Use case: user finds a leaked device	71
5.5	Extending and integrating eReuse.org	72

5.6	Maximize the social usage.....	72
5.7	Providing inputs for circularity.....	74
6	DeviceHub.....	75
6.1	Requirements.....	75
6.1.1	Target users.....	75
6.2	Implementation.....	76
6.2.1	Technologies used, development model and design.....	76
6.2.2	Installation and configuration.....	81
6.2.3	Managing metadata of devices.....	82
6.2.4	Managing users.....	88
6.2.5	Performing events.....	89
6.2.6	Managing places.....	94
6.2.7	Tagging devices.....	94
6.2.8	Interacting through the eReuse.org ecosystem.....	96
6.2.9	Emitting certificates.....	96
6.2.10	Extendable and integrable.....	97
6.3	API.....	99
6.3.1	Login.....	100
6.3.2	Working with devices.....	101
7	Sustainability.....	102
7.1	Social and environmental dimensions.....	102
7.2	Economical dimension.....	102
7.3	Sustainability matrix.....	103
8	Conclusions.....	104
9	Bibliography.....	105
10	Annex.....	112

LIST OF FIGURES

<i>Figure 1</i> The result of an inventory is the metadata of a device in an agent. In this case, the information is about a computer and its components [15].	21
<i>Figure 2</i> Life cycle for a device.	23
Figure 3 Information of a computer and its components [19].	24
Figure 4 Social projects in Reutilitza.cat [15]	26
Figure 5 Gantt diagram of the sprints for the architecture and DeviceHub, with dependencies.	30
<i>Figure 6</i> Human costs per tasks.	34
Figure 7 How the PXE server works.	38
Figure 8 Main page of DeviceHubClient. The design is divided into three rows: a list of places, a list of devices and information for the selected device.	40
Figure 9 Users can perform different actions to multiple devices at once, such as move them to places, generate tags, or perform events such as repairing devices or allocating them to a client.	40
Figure 10 DDI automatically obtains metadata about a computer and its components. This information is then uploaded to a DeviceHub. Users interact with DeviceHub to manage reusing and disposal until collection.	40
Figure 11 Representation of the interface a user sees in the DeviceHub App, when it is performing an action (what we call an “Event”) to a device. We can appreciate how the app shows the position of the user in a map. The position is going to be sent to the server, and it is going to be assigned to the device.	42
Figure 12 Map of collection points of Barcelona, as seen for DCP. The numbers on screen group collection points that are close.	43
Figure 13 Technologies used by the components of the eReuse.org system and their relationships.	43
Figure 14 The eReuse.org architecture with the main components and relationships.	44
Figure 15 Class diagram for Device and types of devices, except Component, which is shown in Figure 16.	48
Figure 16 Class diagrams for Component and its subclasses. Component extends Device, as shown in Figure 15.	49

Figure 17 List of events for a device in a DeviceHub, in ascending chronological order (bottom is first). For this list, we know that the device has some components, and the hard-drive was tested and erased (see below the event Snapshot). After this, the device was set to be Ready to be reused, and finally it was allocated to someone.	50
Figure 18 DeviceHub Class diagram for Event, and events with one device.....	52
Figure 19 DeviceHub class diagram for events with multiple devices.....	54
Figure 20 DeviceHub class diagrams representing Place, Account and Benchmark.	54
Figure 21 An example of how the field url and sameAs work.....	55
Figure 22 An exemplifying preparation for the reuse process, extended from WRAP [42]. Note that identifying can be done at the beginning of the process and can be repeated after refurbishing the devices.	58
Figure 23 The preparer can personalize the labels and preview them in real-time. Once it is satisfied, it can “print” them to a PDF. Thanks to the co-design aspect of the development, we acknowledge that the majority of our users prefer PDF to printing directly, as they can send it to the co-worker responsible for printing, and they can always print the PDF easily.	60
Figure 24 (Left) Screenshot of DeviceHub. We can see the many default events that we can perform on a device. Ready (which is the hovered one), ToRepair, Repair, ToDispose, and toPrepare are exclusive for the preparation for reuse. Their usage depends on the granularity an organization looks for in controlling this process.	61
Figure 25 (Right). Interface of the eReuse.org app, where we can see how the user is creating “Building A”, which contains the painted area in grey, with the grey markers as vertexes, and the blue one as the position of the user. This “Building A” can be a warehouse or the location of one client, for example.	61
Figure 26 Types of transferal by different criteria.....	62
Figure 27. The process of preparation for reuse and transfer by directly donating a device...	63
Figure 28 (Left) Workflow for the devices in the transferal process. These actions are performed in a DeviceHub. The state ‘Ready’ references the final state seen in the last section.	65
Figure 29 (Right) Workflow for the projects in the transferal process. These actions are performed in a TransferHub. Both systems need to cooperate by notifying each other when an event is triggered in a device or a project. For example, WaitingForAssignment changes to DevicesAssigned when a manager performs the event Allocate from some devices to the specific project. This implies that TransferHub needs to share the projects to DeviceHub.	65

Figure 29 Simplified representations of the use case Indirect donation through preparation, and final recycling.	66
Figure 30 Collection process.	67
Figure 32 Diagram representing the technologies of DeviceHub and DeviceHub client, and how they interact with the user and with each other. DeviceHubClient supports the latest versions of Chrome, Firefox, Safari, and Opera.	77
Figure 33 Login screen for DeviceTag.io.	78
Figure 34 (Top) Mock-up representing the main view in. (Bottom) Representation of the right top menu in DeviceTag.io.	78
Figure 35 The three-column design of the sub-view for managing devices.	79
Figure 36 The UI reacts when the user clicks on one of the checkboxes of the device list, showing possible actions.	79
Figure 37 The modal.	80
Figure 38 (Left) When selecting devices, users can apply filters that make already-selected ones disappear from view. To ensure they are still selected, there is a counter of devices that, upon hovering the cursor, shows a list of selected devices.	80
Figure 39 (Right) The preview of a resource (the floating box in the image) is useful when the user wants to check out the resource (in this case information about the Place ‘TXT’), without having to lose the task it was doing.	80
Figure 40 Workflow of a Snapshot, from executing through DDI until it is fired in a DeviceHub. DHC stands for DeviceHubClient.	83
Figure 41 (Left) The interface of DeviceTag.io (DeviceHubClient) for performing login is minimalist and simple.	88
Figure 42 (Right) DeviceHubClient offers at the right-top of the screen a button reminding the user which database it is using, and by pressing it a drop-down menu appears with the different databases to which the user has access.	88
Figure 43 Representation of the workflow of a collection.	93
Figure 36 Setting the area of a place in DeviceHubClient, with the assistance of Google Maps.	94
Figure 37 Selecting a place filters the list of devices, showing only the devices that are on that place.	94
Figure 46 A tag generated by DeviceHub, in concrete by DeviceTag.io.	95
Figure 47 Main options in DeviceHubClient when personalizing the tags.	96

Figure 48 DeviceHubClient generates the forms by translating the information in the schema endpoint to the specification of Angular Formly, and extends this module by providing functionality. In this case, the DeviceHubClient understands that a place needs a list of devices, so it offers a widget to the user to select some devices. It also understands that a place needs a pair of geolocation coordinates, so it adds a widget of Google Maps for the user to select a position.99

LIST OF TABLES

Table 1 Springs with their working hours and descriptions. Sprint ID 3 and onwards are part of the implementation of DeviceHub.....	31
Table 2 Non-human costs for the project.....	34
Table 3 Total cost for the project.....	35
Table 4 Table describing the events regarding devices with the systems that use them. To understand the events fully, we recommend reading the rest of the eReuse.org architecture, where they are presented within their context.	50
Table 5 Aggregated results of the detection and uniqueness of serial numbers of computers, and some of their components. In the case of the RAM module, we only used one randomly from those available in a computer. Nowadays, the percentage of detection and uniqueness has significantly risen as a result of enhancements; however, we can extract the same patterns.....	69
Table 6 Sustainability Matrix, extracted from Christian Felber “The economy of the common good” [58].	103

LIST OF CODE LISTINGS

Listing 1 Example of an HTTP link header representing a Device. Although in the specification, is future work to create the ontology files.	57
Listing 2 Example of a response containing the error <code>UnauthorizedToUseDatabase</code> from DeviceHub.	57
Listing 3 Simple regex representing the HID (it could be more precise).	70
Listing 4 Example of HID, representing the computer Acer with model aod270 and serial number lusga_0d0242201212c7614-aod270	70
Listing 5 Basic instantiation of DeviceHub.	81
Listing 6 Recommended minimum configuration for DeviceHub. It sets two databases, the own URL, the URL of DeviceHubClient, and an account for the GRD module (see Section 6.2.8), as this module needs an account to access DeviceHub (which is created if it does not exist).....	81
Listing 7 Configuration object for DeviceTag.io.....	81
Listing 8 Apache2 configuration for DeviceHubClient. Note that you may change the port, the server name and the two paths.	82
Listing 9 Illustrative Snapshot.	83
Listing 10 Two real examples of HTTP requests with a prefixed database.	88
Listing 11 Exemplifying GET query filtering results. In this case, it obtains all events from the public database of DeviceTag.io that are snapshots. It is using MongoDB syntax, so it supports the majority of filters MongoDB uses.	100
Listing 12 Exemplifying GET query ordering results. In this case, it obtains all devices from the public database of DeviceTag.io, sorting them by the fields of byUser (ascending) and byOrganization (descending). It is using native Python syntax.....	100
Listing 13 Exemplifying GET query embedding results. In this case, it obtains all devices from the public database of DeviceTag.io, and embeds the user object, and a list with all the objects of all events, using MongoDB syntax.	100
Listing 14 Exemplifying GET query with projection definition. In this case, it obtains all places from the public database of DeviceTag.io; however, it does not retrieve the field byUser. It uses MongoDB syntax.....	100
Listing 15 Exemplifying GET query combining different operations. In this case, it obtains all devices from the public database of DeviceTag.io that are TFT (sub type of monitors) and it sorts them by labelId (ascending).	100

Listing 16 Exemplifying a login request.....	100
Listing 17 Successful login response.....	100

1 INTRODUCTION

Many digital devices from businesses and public organisations are dismantled and recycled when amortized or out of guarantee, despite more than 80% being nearly up-to-date and in perfect condition [1].

The latest research on e-waste estimates that about 41.8 million metric tonnes (Mt) of e-waste was generated in 2014 and that this number will increase to 50 Mt as early as 2018 [1]. In Europe, the total e-waste generation was 9 Mt in 2012 and 11.6 Mt in 2014, where in 2012 only 3.2 Mt of e-waste was officially collected.

Even though it is a paradox that the volume of e-waste is growing three times faster than other types, recycling plants collect less and less e-waste. This is mainly attributed to pillaging at collection points or by home collection done by unauthorized operators [2]. A research undertaken by the Countering Waste Electric and Electronic Equipment (WEEE) Illegal Trade (CWIT) found that from 9 Mt of WEEE in Europe, only 35% ended up in the officially reported collection and recycling systems, and the remaining 65% was: 24.39% exports (of which 86% were illegal exports), 51.22% recycled under non-compliant conditions in Europe, 12.2% scavenged from valuable parts, and 12.2% thrown in waste bins [3].

The current practices for dealing with locally generated WEEE seem unsatisfactory, because they lead to a loss of secondary resources and damage the environment [4]. Third countries receive waste from developed countries: more than 90% of discarded computers from the developed world is exported to developing countries such as Ghana, Pakistan, and India [5]; where the waste is poorly collected and improperly disposed. 59% of waste is not collected [6] and typically ends up in informal waste dumps.

Finally, the digital divide¹ undermines development in third-world countries: 59% of households worldwide do not have computers, and 63% do not have Internet access [7] (83% in third world countries in 2007 [8]).

There are alternatives to mitigating the production of e-waste: reduction and reuse.

If the reuse of digital devices ensures recycling, then effectively contributes to: generating a circular economy², preventing waste generation, reducing the risk of WEEE issues such as leakage to landfills or illegal exports, creating jobs, and strengthening digital skills. Reuse can also help to reduce the digital divide and strengthen institutions and projects for social change [9]. However, why is it such a minority practice in Europe? There are different factors:

- When companies, governments, or individuals need to get rid of their digital devices and they want them to be reused, they do not know where to turn. This results in most

¹ The digital divide is the divide between those with access to new technologies and those without [60].

² “Circular economy is a global economic model that aims to decouple economic growth and development from the consumption of finite resources. It is restorative by design, and aims to keep products, components and materials at their highest utility and value, at all times. Unlike a linear economy, it is about optimising systems rather than components. This includes careful management of materials flowing in both biological and technical cycles” [10].

Electric and Electronic Equipment (EEE) being recycled too early, despite the explicit demand for reuse coming from social and charity organisations.

- Without a traceability solution, which uniquely identifies digital devices and their components, and a commitment from consumers of reused products, it is uncertain if donated digital devices for reuse may end up being exported illegally and may potentially pollute the environment. Such risk and slippage is the main drawback in the promotion and the practice of reuse in public and private organisations or individuals.
- The lack of standards, studied processes, and technology for reusing results in costly operational costs and reduces benefit margins, which leads towards unsustainability.

Finally, the circular economy cannot be guaranteed without a traceability solution, commitment of the stakeholders, and correct monitoring through indicators that measure circularity: the life cycle of the devices through reuse and final recycling. Inputs need to be gathered about the durability of devices and their components, the intensity of their usage, information about repair and maintenance, and how much is ultimately collected for recycling, among others [10].

To solve these problems, the eReuse.org project proposes a “set of open-source tools, open models and services to support reach circular economy of electronics by increasing lifetime, and by ensuring traceability until recycling. eReuse.org envisions to empower and engage people around the world to create local communities which bootstrap electronic reuse, and to support the development of a globally recognized reuse quality and traceability standard. Any initiative focused on fixing, reusing, recycling, and in general promoting zero waste on electronic equipment are warm welcomed; we transfer to all of them business models, services and technologies to gain profitability, effectiveness, efficiency, and traceability to ensure final recycling” [11].

Therefore, for reuse to succeed, there is a need for services and technologies to gain efficiency and traceability [1] [11] [12]. Then, this project’s research question is: How can reusing be efficient, secured, and analysed to ensure final disposal? This project proposes the design of the eReuse.org architecture, defining a set of open-source tools and implementing one of those: DeviceHub, an IT Asset Management System focused on reusing.

1.1 OBJECTIVES

The main technical objective for eReuse.org is to propose a software solution to:

1. Efficiently manage reusing and disposal until collection for recycle.
2. Contribute to ensuring the reuse and recycle processes by supporting traceability.
3. Facilitate third parties to join eReuse.org’s effort by using, extending, and integrating its system.
4. Maximize the performance of devices and their usage-time.
5. Generate input for the indicators that measure circularity, and access to them.

1.2 STAKEHOLDERS

We identify the following groups of stakeholders:

- Givers: users or organizations initially owning the devices, who are willing to transfer them, by donation or sale. Givers are interested in knowing the state of the devices they give, knowing if they are treated legally, and following their policies.
- Receivers: final users of the devices, who are interested in the reused or simply cheaper ones. They buy or get them as donations, and they are in charge of their disposal.
- Analysts, auditors, collaborators: usually volunteers, organised in thematic communities (web portals), concerned with aspects such as environmental sustainability, governance of the resource system, software development, repairing, protection of consumers' rights, protection of citizen's rights, and protection of the environment. Volunteers can contribute to identifying and promoting contributions (registering devices into the system), management, and allocation of digital devices to future users according to needs or social support.
- Public administration: interested in managing specific attributions and obligations to regulate the participation of society, monitoring environmental effects and data protection, and satisfying their own needs regarding devices.
- Providers: professionals or volunteers that provide services to achieve reusing and recycling.
 - a. Preparers: they manage the preparation for the reuse. They are required to ensure the givers' demands described above; at the same time, they focus on efficiency in preparing the devices.
 - b. Selling channels: from social platforms to second-hand shops and distributors, they manage transfers. Their objective is selling or donating the devices from the preparers to the final users.
 - c. Transporters.
 - d. Waste managers: certified entities to work with waste. They can be collection points, in case they get disposed devices, such as stores receiving devices from the final user or reuse and recycling centres.
 - e. Repair cafés: organizations that help and teach users how to refurbish, repair, and upgrade their devices.
 - f. (Re-)manufacturers: interested in producing or selling new or remanufactured devices.

1.3 SCOPE AND REQUIREMENTS

This project designs the software architecture of eReuse.org, which efficiently manages the reuse process ensuring collection, and develops and implements one of the components of the architecture: DeviceHub, an IT Asset Management System (ITAMS) focused on reusing and recycling devices.

We present the set of requirements for the software architecture, which we analyse in Section 5. In Section 6.1, we present the specific requirements for DeviceHub.

1. To efficiently manage reusing and disposal, by proposing a set of software systems, ontology, and protocols. Focusing on:
 - a. Managing the preparation for the reuse: identifying and refurbishing devices.
 - b. Managing the transferal and any other process involved in allocating and giving the device to the final user.
 - c. Managing the collection, when devices are disposed and taken to a collection point, usually for being recycled.
2. To automatically recollect, process, analyse, control, and share metadata about devices to guarantee traceability. The metadata can be from the approximate location of the device, to the characteristics and serial number of its components.
3. To manage places involved in the life cycle of devices, supporting traceability. For example, it is interesting to know where to pick up a device when transferring it or the waste dump a device is found.
4. To securely and uniquely identify a device and its components around the world.
5. To maximize the social usage of devices, by helping in matching the best potential users for specific devices or in improving the usage-time.
6. To offer inputs (open data) for the indicators that measure circularity, cleaning the data by removing any private information, and aggregating it, avoiding performance costs to auditors and making the information accessible.
7. To offer an ontology that is easily understandable, replicated, and extended, so stakeholders can use, integrate, and extend the architecture.
8. To maintain the privacy of the users and their devices.

1.4 LIMITATIONS AND RISKS

The project defines the eReuse.org architecture. Although it introduces the theory to justify the technological decisions, it is out of the scope of this project in researching the viability and real necessity for the software developed.

This documentation is not the description of a finished project, but the definitions, methodologies, and implementation of a project that we expect to hit full release in the following months. Everything is susceptible to change; however, we expect it to be minimal.

We do not detail everything about the eReuse.org architecture or DeviceHub, as the objective of this documentation is not for people to be able to reproduce the project in full, but to guide them through the main decisions, so they can understand, use, and extend the project and integrate it with their systems. Documenting the project in full detail would be difficult for stakeholders to understand, and it would need a dangerous amount of time.

There are many risks in the project. The greatest are:

- Adoption: organizations not adopting eReuse.org because of the technology or because they do not want to invest enough in reusing and recycling. For the former, although this project is being co-designed and co-developed with stakeholders, new ones can introduce barrier requisites too costly to develop. The latter is a problem of the actual political and economic contexts: although corporate social responsibility

(CSR) strategies and circular economy are gaining momentum, they may not be important enough for many organizations to justify investment.

- Recognition: eReuse.org proposes a standard for tracking devices, so they can be reused and recycled correctly. Standards require acknowledgement for many third parties, and achieving this is an enormous and difficult task.
- Technological: problems due bad design decisions or bad programming approaches may result in costly fixes or refactors. We talk more about it in the deviations subsection of the planning.

2 STATE OF THE ART

The section is divided in five parts: (1) definitions; (2) justification, supporting the need of this project; (3) technical background, explaining the prototype and the need to split in new projects; (4) market evaluation, showing main similar existing solutions; and (5) final conclusion.

2.1 DEFINITIONS

2.1.1 Reuse and recycle

Reuse means using an item again after it has been used, without any structural change to its materials. In opposition, recycling changes its material structure, so it is no longer considered the same item.

2.1.2 Device

Device is a meticulously chosen word by its generic definition, which represents our purpose to be able to reuse almost anything, although we start with a subset of electronic devices.

2.1.3 IT Asset Management System

According to Gartner, “IT Asset Management (ITAM) entails collecting inventory, financial and contractual data to manage the IT asset throughout its life cycle. ITAM depends on robust processes, with tools to automate manual processes” [13]. Then, an IT Asset Management System (ITAMS) is the system that helps with managing IT assets. An ITAM strategy or program can be composed of the following aspects [14]:

- IT acquisition management: the management that starts from the identification of the necessity of an asset until it is bought. The objective is to acquire the IT assets that are needed for the organization, in the most effective and productive way.
- Asset identification management: the process of extracting the metadata of an asset for identifying it, and registering it into the system.
- Policy management: sets the protocols towards the IT assets of the organization.
- Financial management: keeps control of the expenses, contracts, and other financial data regarding the IT assets.
- Communication and education management: responsible for all communication and education initiatives within the ITAM program.
- Legislation management: involves the monitoring of all government regulations that impact the organization and can be supported by the ITAM program, resolving non-compliance threats and risks, for example.
- Compliance management: conducts all compliance activities involving the ITAM Program, with the objective to reduce risks and the ability to audit responses.
- Documentation management: manages all documents associated with IT assets, such as manuals, version control, etc.
- Disposal management: defines the processes for organizations to remove IT asset management from their environment, and provides ways for doing so.

- Program management: provides centralized procedures, cost saving strategies, accountability for compliance, tools, and tracking for all ITAM related efforts.
- Project management: defines those characteristics of a project team that are essential to successfully conducting the initiatives that are needed to meet the goals of the ITAM program.
- Vendor management: implements a “fair and consistent” enterprise-wide strategy to develop and manage vendor relationships.

In our case, DeviceHub is a system to manage devices (a subtype of asset), focusing on reusing them. We need to comment that we do not incorporate financial and contractual data as an objective for this project, but as this information is useful to generate indicators about circularity, it can be future work.

2.1.4 Inventory

Inventory is the action of uploading the metadata of a device, such as traceability aspects, tests performed, and hardware identifiers to an agent.

The screenshot shows the website reutilitza.cat with a navigation bar and a search bar. The main content area displays the metadata for a device with ID 6502. The metadata is organized into several boxes:

- CPU:**
 - Name: Intel(R) Core(TM)2 Duo CPU E6550 @ 2.33GHz
 - manufacturer: GenuineIntel
 - CPU speed: 2.33
 - Speed measure: GHz
 - number of cpu: 1
 - number cores: 2
 - CPU score: 9308
- RAM:**
 - RAM size: 2
 - ram measure: GB
 - used slots: 2
 - Free slots: 2
- 1st graphic card:**
 - Model VGA: Intel Corporation 82Q35 Express Integrated Graphics Controller
 - Memory: 512
 - VGA measure: KB
- 1st hard drive:**
 - size hdd: 74
 - hdd measure: GB
 - ansverson=
 - interface hdd: SATA
- Audio 1:**
 - Model audio: 828011 (ICH9 Family) HD Audio Controller
- Net 1:**
 - Net speed: 1Gbit/s
- Brand:**
 - Dell Inc.
- 1st unit:**
 - Unit model: DVD reader

Figure 1 The result of an inventory is the metadata of a device in an agent. In this case, the information is about a computer and its components [15].

2.2 JUSTIFICATION

To justify the project, we first present the device life cycle, as a way to contextualize the main processes involved in reusing and recycling; we analyse if existing software solutions can be used for managing such processes; we discuss the viability of using the actual prototype;

and, finally, we present the conclusions, justifying the need for creating the architecture and DeviceHub from scratch.

2.2.1 The life cycle process

The digital device life cycle is an accumulation of stages a device goes through from its creation to its final destruction. Digital device life cycle can be linear if a device is used just one time and destroyed afterwards, or circular if it is reused more than once [1]. This concept is tight with circular economy, as it proposes a circular life cycle [10]. Note that, to reach a correct life cycle, the device, at the end of its life, needs to be pre-processed (dismantled) and post-processed to recover resources and segregate hazardous waste.

The life cycle of electronics, depicted in Figure 2, essentially contains these phases: production, manufacturing, transfer (retail, donation), use, collection, recycling, and preparation for reuse. We have four loops divided into two groups:

- The waste loops consist of *recycle* and *waste to product* loops. They start when a product reaches a collection point, becoming waste, and it can only return to be a product if it is prepared under a certified process.
 - Recycle loop: is the traditional loop where products decompose and are built again. Raw material is introduced in the loop.
 - Waste to product loop: waste is refurbished and prepared for reuse. This loop does not consume raw material, and mainly has agents such as reuse, recycling centres, and collection points.
- The product loops consist of *product transfer* and *personal* loops. They aim to keep products and components alive as long as possible, performing basic preparation for reuse consisting of repairing, upgrading, or transferring by reselling or donating to avoid waste.
 - Product transfer loop: the product is prepared and then transferred to the final user. eReuse.org proposes direct transfer from giver to receiver, without intermediaries, reducing management and cost; it is not always achievable if the device is broken or outdated, as both users may not necessarily need or want to repair or upgrade it [1].
 - Direct loop: the original user reuses the device, after upgrading it, repairing, or installing a more lightweight Operative System (OS). This is the most efficient way of reusing, as it requires the least possible steps and intermediaries.

At any time of the cycle the device can be leaked, because it has been lost or stolen. In this case, the device can be exported to other countries, treated without permission or ending up in landfills.

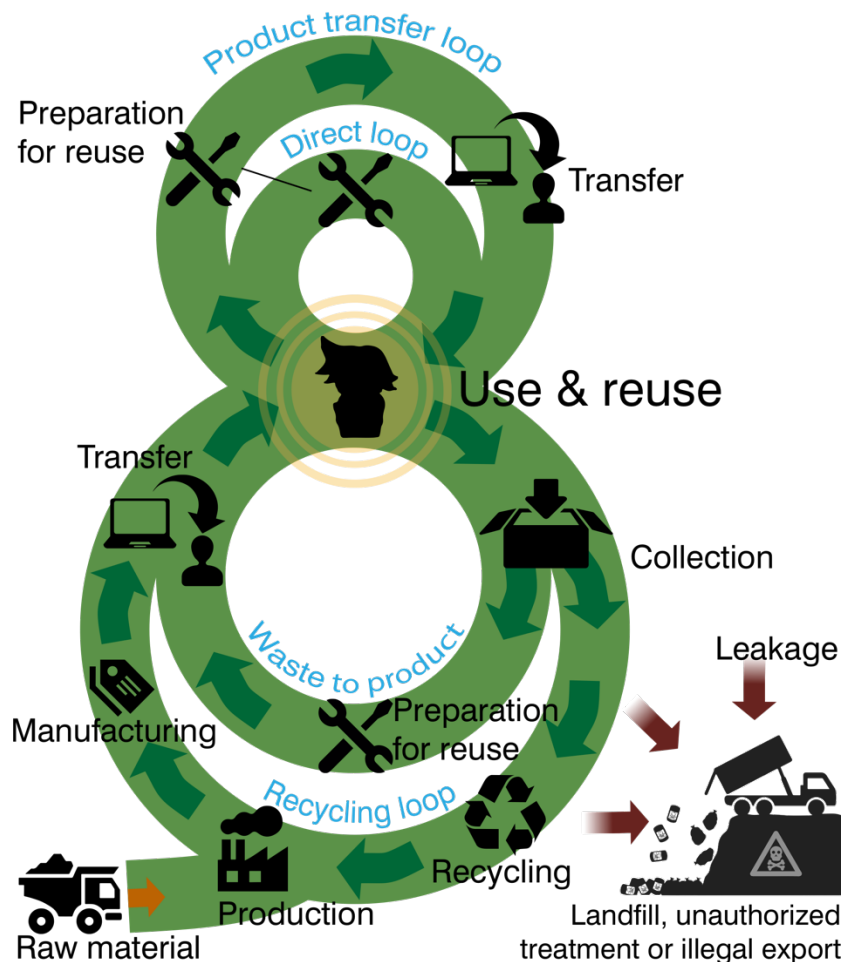


Figure 2 Life cycle for a device.

2.3 EXISTING EXTERNAL SOLUTIONS

Currently, there are many local initiatives focused on reusing, such as labdoo.org, reutilitza.upc.edu, computeraid.org, and promiseit.ie. Some of them are corporate donation programs that facilitate companies' secure donation of IT equipment to schools, charities, and community groups. The main problem with these initiatives is that they are rare in Europe and cooperate little between them.

For the components of the architecture, at the moment of writing, there are no collaborative solutions (in the sense of systems of different organizations cooperating) with the main focus of reusing and recycling. However, we can find existing solutions that offer similar functionality than the components eReuse.org offers, such as the ITAMS DeviceHub. We present already-existing ITAMS, stating their strength and weaknesses.

2.3.1 GLPI

GLPI is a PHP and free open-source IT Asset Management System from 2003 with an additional administration interface [16]. It is one of the main software of this kind with a big community that has developed many functionalities [17]. GLPI's idea is to be the main piece in asset management, and has extensions that interact with systems that offer other functionalities. For example, GLPI relies in third parties (called agents) for automatically

getting information from computers. The system general characteristics are ³ [18]: (1) organizational structure management; (2) user management; (3) permission system; (4) searching; (5) notifications; (6) inventory of computers with management of its components, disk space and TCO management (see Figure 3); (7) inventory of peripherals such as monitors, printers, and external devices; (8) inventory of software; (9) inventory of hardware by geographic area; (10) administrative and financial information management; (10) management of various states for materials; (11) history of modifications of elements; (12) tracking requests; (13) project management; (14) business rules; (15) managing technical interventions to devices; and (16) managing of employees.

Components	
1x	System Board AW8-MAX Chipset: chipset 888
1x	Processor Athlon 64 FX-55 Frequency: 1247 Frequency: 1391
1x	RAM CM2X256A-5400C4 Type: EDO Frequency: 234 Size: 982
1x	Hard Drive Deskstar T7K250 Rpm: 4634 Interface: IDE Cache: 1878 Capacity: 43999
1x	Network Card DFE-538TX Flow: 164 Mac Address: 8:0:14:1e:28:45
1x	Drives DRW-1608P Writing ability: Yes Speed: 58

Figure 3 Information of a computer and its components [19].

2.3.2 Ralph

Ralph is an “Advanced Asset Management, DCIM and CMDB system for data centre and back office.” It is an active project from 2011 made by the Allegro Group, and written in Python, using Django and Redis. Its main characteristics are: (1) keeping track of assets purchases and their life cycle, (2) flexible flow system, (3) data centre and back office support, and (5) data center visualization built-in [20].

2.3.3 OCS Inventory NG

OCS Inventory (Open Computers and Software Inventory Next Generation) is a technical management solution for IT assets; it was made in 2001 and held by the OCS Inventory Development Team. It “is an application designed to help a network or system administrator

³ Note that this is a non-complete list with what we consider the most important elements. Please, refer to the source for a complete list.

keep track of the computers configuration and software that are installed on the network.” It can work with GLPI and is free and open-source software released under the GNU General Public License, version 2.0 (GNU GPLv2) [21].

2.3.4 Spiceworks

Spiceworks is a free ITAMS (not open-source) created in 2006 and managed by Spaceworks Inc. Its main functionalities are: (1) help desk with ticket management, (2) network monitoring of devices, and (3) inventory of devices through network [22].

2.3.5 Landesk

Landesk is proprietary paid software made by LANDESK Software focused in offering a unified experience through IT service management, IT asset management, systems management, security management, and enterprise mobility management. Its main characteristics are: (1) discover and invent assets through the network, (2) policy and end user management, (4) software license management, (5) OS provisioning and migration (mass installation of operative systems through the network), (6) software distribution through the network, (7) alerting and monitoring devices through the network, (8) reporting about assets, and (9) power management [23].

2.4 TECHNICAL BACKGROUND

We present a technical background explaining the actual prototype.

Reutilitza.cat is a social, charity and not-for-profit platform with the aim to provide unused digital devices to social entities and to the most excluded sectors of population [24]. Reutilitza.cat is powered by the prototype⁴, initially developed by Fernando Ramírez Lázaro, Sergio Caballero Peña, and José Antonio Giner; directed by David Franquesa, Leandro Navarro and David López [25] [26] [27].

XSR was created on the 21st of December in 2010 because of the necessity of UPC-Reutilitza⁵, as they had to manage an increasing number of computers. The project had the objective of “developing a Web 2.0 application with Drupal to make the process of donating/receiving computers and the exchange of knowledge between people and social organizations much easier” [27]. XSR became Reutilitza.cat, which is a website where donors can upload metadata about their device, and a participatory social process, with the consent of the donor, would select a social project to receive the device. Receivers can create social projects, which are shown on the website and can be voted on by anyone as a way of getting social recognition, resulting in more chances to get devices (see Figure 4) [15].

The prototype has been co-designed with stakeholders, adding more functionality mainly alienated with the objectives of this project. Many of these objectives could not be fulfilled,

⁴ At the beginning the prototype was called *Xarxa de Suport a la Reutilització* (XSR, Reutilization Support Network).

⁵ UPC-Reutilitza is a program from the association of TxT (Tecnologia per Tothom), which mainly consists of “repairing and updating computers donated by companies, which will be received by social service entities” [70].

and Reutilitza.cat is working under an incomplete prototype. The main technological difficulties are:

- Drupal 6, the used framework, is from 2008, which means it has outdated technologies and mechanics that today are far lesser time consuming to develop and maintain [28].
- The framework is ending its support [29] and updating to the new version is complex, as there are many deep architectural changes [30]. By ending its support, it means that the site will not receive any update to fix errors, implying, for example, that it will be vulnerable to attacks.
- The incremental designing and developing means taking technological decisions that are removed later or needed to be adapted, which results in cluttered code that is difficult to maintain and develop.
- The system is not thought to be integrated with others. The main way of accessing the website is through REST using HTML, such as a browser, which is unpractical.
- The project contains much functionality that is suitable for more focused, divided projects, instead of a big monolithic one.
- As a prototype, the code is not polished per se.

Social projects



[See more projects](#)

Figure 4 Social projects in Reutilitza.cat [15]

2.5 CONCLUSION

There is no software specifically made to manage the life cycle of devices. There are actual solutions that could offer part of the functionality (they only manage devices inside the organization without taking care of the life cycle), and we could build our solutions on top.

Landesk and Spiceworks (and any other privative solutions) are automatically discarded because they are not open-source and free; it would represent a great barrier for adoption for

organizations that cannot handle the cost, and it goes against our philosophy. The rest of the open-source solutions, although offering some needed functionality, have several problems: (1) they do not ensure our required precise identification of devices; (2) they do not cooperate (there are no *core* built-in mechanisms to share information between organizations); (3) based in our experience, GLPI and Spiceworks are difficult for beginners, as they overwhelm them with many options and configurations, a result of all the functionality they have; and (4) changes to fulfil the requirements need transversal interventions through the different layers of the existing systems, and the majority of them are big projects and some even old (GLPI is from 2003, OCSInventory from 2001).

As a conclusion, there is no system that provides the functionality eReuse.org needs. The prototype has many difficulties that make further development unsustainable, and building on top of an existing one has many difficulties. This is why we propose building a solution from scratch, a system focused only in managing the life cycle of devices, complying with the set of requirements without complications.

In the case of the component being developed, DeviceHub, this results in focusing on a tool to easily and efficiently manage the life cycle of devices, but also being able to integrate with GLPI and the other solutions described (future work might be developing such integrations), as we understand that it is unsustainable to surpass them in functionality, and then expect organizations to migrate everything from their already-existing ITAMS.

3 METHODOLOGY, TEMPORAL PLANNING, AND TECHNICAL COMPETENCES

3.1 METHODOLOGY

This project adopts Scrum methodology.

Scrum is an Agile methodology focused on innovation and based in self-managed workgroups. Scrum is divided in *sprints*, small deadlines covering from two to four weeks. Every sprint is finished with a deliverable. This deliverable is analysed and then appropriate changes are made [31].

As the timetable is constrained and the project is complex, an agile method such as Scrum is a viable option. Small deliverables are a good strategy for co-designed projects such as eReuse.org. We use the task manager Wrike [32].

During the first sprints, as there is not a minimum viable product, meetings are with the co-directors of the project and a few stakeholders. In the last sprints, after the project reaches a milestone and has enough entity to be tried by end users, they are analysed with many stakeholders. The evaluation is going to be with the director and co-directors of the project, and the stakeholders if they are available.

3.2 CALENDAR

The project started in July 2015 and the first open release is previewed for the 4th of July of 2016, where we will release an open version, ready to be used by the public. We will work approximately 1,146 hours, around 25 hours per person per week (this timing will change in some moments to allocate the needed time).

3.3 ORGANIZATION OF THE PROJECT

The project is being developed by Xavier Bustamante Talavera. He performs the roles of project manager, analyst, programmer analyst, architect, programmer, and tester. David Franquesa and Leandro Navarro are the directors and act as product owners.

Xavier Bustamante Talavera (1992) is a computer-engineering student at the Universitat Politècnica de Catalunya (UPC). He is currently the information systems (IS) manager and a developer in the eReuse.org project, where he started in 2013 as a developer. Aside from eReuse.org, he was the president of the Junior Enterprise JEDI⁶ from 2012 to 2014. He has also delivered classes on different web technologies, such as PHP, HTML5, JavaScript, and CSS3. In September 2015, he co-authored a paper, entitled *Breaking Barriers on Reuse of Digital Devices Ensuring Final Recycling*, which has won the Best Paper award at Environinfo 2015 conference held at the University of Copenhagen, Denmark. He considers himself an entrepreneur who enjoys doing projects that help shape a better world.

⁶ Joves Estudiants D'Informàtica, Young Students of Computer Engineering [76].

3.4 PLAN

We divide the project into two phases, following the Scrum methodology.

The first phase is called *Sprint 0*, which started at the beginning of July 2015 and finishes in mid-September 2015. In this phase, we analysed the requirements for the architecture and existing solutions; we defined the basics for the projects that use the architecture, such as DeviceHub, so the team can start developing them; finally, we set an initial plan for DeviceHub and the components.

The second phase englobes the rest of the *sprints*, started after the first one and finishing at the beginning of July 2016 —finishing the project. In this phase, we implement DeviceHub. Meanwhile, we further develop the architecture. The second phase is divided as follows:

1. In mid-March, we released a minimum viable solution, and we implemented it into our production servers, using the trademark of DeviceTag.io (separating the technical open-source project from our cloud service).
2. In July, we will release an easy-to-install and complete solution (which is the one we describe in this project), so third parties can adopt and deploy the technology on their own. Note that this is not the final version; however, a stable and auto-sufficient one.

Each sprint is developed, starting from sprint 1, in the following way: (1) we plan the sprint; (2) we review and update the architecture in key areas for the sprint; (3) we develop a self-contained solution; (4) we develop and run a suite of tests, which the solution needs to pass; (4) we perform a final review with the director and the co-director of the project, and with other interested stakeholders; and (5) we update the documentation. All sprints of DeviceHub provide a graphical interface for them to be executed so non-technical users can successfully review the work done.

Figure 5 and Table 1 represent the main sprints for the development of the architecture and the implementation of DeviceHub.

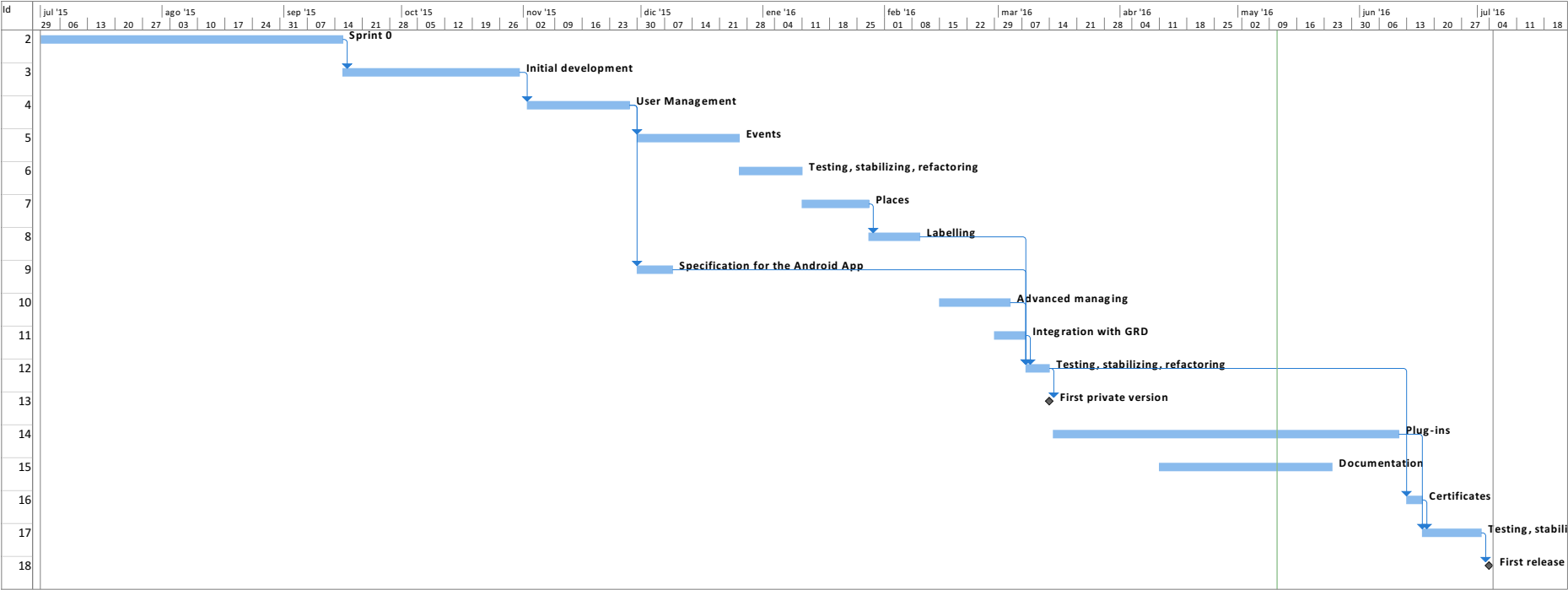


Figure 5 Gantt diagram of the sprints for the architecture and DeviceHub, with dependencies.

Methodology, temporal planning, and technical competences

Table 1 Springs with their working hours and descriptions. Sprint ID 3 and onwards are part of the implementation of DeviceHub.

ID	Task	Working hours	Description
2	Sprint 0	140	Definition of the project and the architecture.
3	Initial development	165	Starting development for DeviceHub, with the objective of creating a proof of concept, and the basis for future development.
4	User Management	105	To provide a basic permission system integrated with the ability to create users.
5	Events	105	To build the full event system, managing devices with simple use cases.
6	Testing, stabilizing, refactoring	65	Stabilizing and improving code, refactoring as needed.
7	Places	70	To create a simple solution to manage the places where reusing and recycling happens.
8	Labelling	50	To produce a tagging system so devices are easily labelled.
9	Specification for the Android App	40	The eReuse.org App is one of the components more integrated with DeviceHub, we define it and adapt code in DeviceHub for it.
10	Advanced managing	75	Adding more options to manage devices and users.
11	Integration with GRD	35	The integration with GRD requires code to trigger connections to it, translating data as needed.
12	Testing, stabilizing, refactoring	30	Stabilizing and improving code, refactoring as needed, performing usability tests, improving design, etcetera.
13	First private version		The first milestone for the project, resulting in code ready for use in our controlled servers. Installation, for example, is not easy and requires high knowledge to touch the system.
14	Plug-ins (extensions)	82	Ability for the system to execute code without being heavily coupled, outside of the main project, in the easiest way.
15	Documentation	99	Generating this documentation (and code to generate it automatically).
16	Certificates	25	Generate reusing related certificates.
17	Testing, stabilizing, refactoring	60	Final refactoring, testing, usability tests and design changes prior to a first public release.
18	First release		First public release. Code is easy to install, use, and extend to third parties.

3.5 DEVIATIONS

There were many deviations in the project, mainly because requirements and priorities change, new requirements are added, we are not experts in the technologies we use, and because such technologies are fairly new so they evolve at the same time we develop the project.

Requirements and priorities change because of the input of stakeholders, and the introduction of new stakeholders. None of the changes were deeply significant, thanks to an acquired experience with our prototype; however, many required modifying the approach of achieving some things. An example of this is generating tags. As our initial idea was directly printing from the browser, many stakeholders noted the necessity of generating a PDF file instead, as the employee using such functionality is not the same that has control over the printer, so the former one needs to send an e-mail to the latter in order to print it.

New requirements were added after welcoming new stakeholders. The most significant addition to DeviceHub is a plug-in system (task n°14), so stakeholders can develop software themselves when their needs are not common enough to be added into the *core* of the DeviceHub project.

The architecture uses fairly new concepts such as linked data or JSON-LD, and DeviceHub uses technologies such as Eve and Angular, to which we are beginners. The added learning curve was an already planned overhead; however, it has resulted to be steeper. This resulted in longer tasks.

Finally, other deviations were caused by bugs and other difficulties; however, they were not infamous enough, and the usage of Scrum made them easy to absorb.

3.6 TECHNICAL COMPETENCES

- CES1.4 Develop, maintain, and assess services and distributed applications with network support (in profundity): The different proposed components of the eReuse.org architecture collaborate through the Internet to achieve specific goals, such as in is DeviceHub.
- CES1.5 Specify, design, implement, and assess databases (a lot → moderate): DeviceHub uses different MongoDB databases, which are automatically configured by the framework Eve when configuring the API endpoint. Finally, we are developing a module using the Aggregation Framework of MongoDB as a way to provide data warehousing. Our expectation at the beginning of the project, according this aspect, has proven higher than what we needed in the end (thanks to the framework).
- CES1.5 Define and manage the requisites of a software system (a bit): we have found and developed many requisites of both the architecture and DeviceHub. However, it is not the intention of this project to present a formal description of the requisites (use cases...), keeping the development of this aspect *a bit*.
- CES2.2 Design appropriate solutions in one or more application domains, using methods of software engineering integrating ethical, social, legal, and economical

Methodology, temporal planning, and technical competences

aspects (a lot): the objective of the project is to come with a software solution to help solve ethical, social, and economic aspects.

4 BUDGET

In this section, we describe the budget with all the involved costs.

4.1 HUMAN COSTS

The duration of the project is approximately 1146 hours. We set a salary of 10€/hour for a student, following a standard Spanish Educational Cooperation Agreement contract for an undergraduate student. Therefore, in total, the human cost of the project is €11,051.66.

In Figure 6, we can see the cost divided per task, which is computed by the number of hours a task requires multiplied by the unit cost.

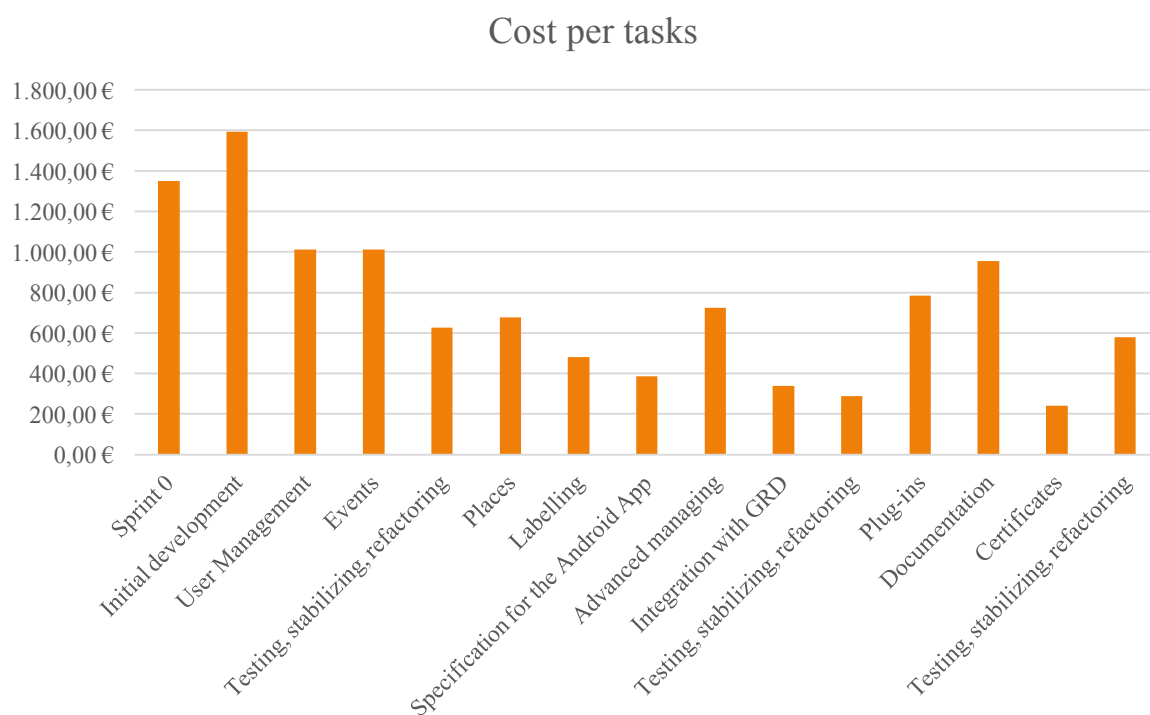


Figure 6 Human costs per tasks.

4.2 NON-HUMAN COSTS

We compute hardware and software costs by establishing amortization in 4 years, and using them 8 hours per day on all the regular working days. In the eReuse.org project, they set a 75% overhead over the salary of a person for basic needs (electricity, water...), administrative costs, and other expenses. We take the same percentage over the final human-cost of the project. See Table 2 for details on the non-human costs.

Table 2 Non-human costs for the project.

Hardware			
Name	Price	Units	Cost

Personal Computer	€700.00	1	€ 52.71
Android smartphone	200.00	1	€15.06
Total cost			€67.77
Software			
Product	Price	Units	Cost
Windows 10 Pro	€279.00	1	€21.01
MS Project Professional 2016	€1,369.00	1	€103.09
MS Office Professional 2016	€539.00	1	€40.59
PyCharm	-	1	-
Webstorm	-	1	-
Total cost			€164.68
Services			
Service	Price/month	Months	Cost
GitHub	-	8	-
Hosting	€20.00	7	€140.00
Total cost			€140.00
Basic needs, administrative work...			
Human costs			€11,051.66
75% of human costs			€8,288.75
Total cost			€8,661.20

We use a PC, Windows, and Microsoft Office to document and develop. We may use an IDE as PyCharm and Webstorm to develop, although it depends on the final technology we use. As for the services, we need Github to store, track, and share the code, and a host to run it (the price of the hosting is an estimation, as it depends on the technology we use).

The only non-human cost that is tied to a task is the Android Smartphone, which adds €20.08 to the task *Android App*.

4.3 RISKS

Some minor risks can extend the duration of the project. We compute 15 days of extra work and an additional 5% of risk to the total cost.

4.4 TOTAL COST

The total cost by developing the project from scratch is €21,486.00.

Table 3 Total cost for the project.

Human costs	€11,051.66
Non-human costs	€8,661.20
Extra days	€750.00
Risk at 5%	€1,023.14
Total cost	€21,486.00

4.5 CONTROL MANAGEMENT

Human costs depend on time; therefore, these can increase if the amount of time to accomplish the tasks increases. This is why we compute the hours we work in every task. At the review phase of every sprint, we update the time and re-compute global costs. We have an indicator to know how much time we have delayed (or advanced) the project, being the result of subtracting the date of the expected deadline with the date of the definite deadline.

The non-human costs are products or services that we do not need to repair if they fail. They cannot produce a noticeable delay in the project if they do so for some time. Thus, we will not control them.

5 EREUSE.ORG ARCHITECTURE

The eReuse.org architecture defines a distributed multi-agent system⁷, instantiated by multiple organizations and implemented as software services, tools, and data repositories, with the main objectives of: (1) managing reusing and recycling, (2) contributing in ensuring reuse and recycle by supporting traceability, (3) facilitating third parties to join eReuse.org's effort by using, extending, and integrating the system, (4) maximizing the social usage of devices, and (5) generating inputs for the indicators that measure circularity. The multi-agent system is heterogeneous, so it is composed of different types of agents; cooperative, as the agents work together to achieve the main goals; has a set of common knowledge; and is in a dynamic (nonstationary) environment, such as the Internet.

This section is divided as follows: we present the different technological components the system is made of, the general definitions for the ontology and the API, and finally how the architecture achieves every objective.

5.1 COMPONENTS

The main components of the eReuse.org system are Device Diagnostic and Inventory (DDI), DeviceHub, the Global Record of Devices (GRD), TransferHub, the eReuse.org App, and the Directory of Collection Points (DCP). Figure 13 and Figure 14 respectively detail the schema and the used technologies.

5.1.1 Device Diagnostic and Inventory (DDI)

Device Diagnostic and Inventory (DDI) is a set of tools that supports the process of preparing for reuse. DeviceInventory is provided in a live disk Ubuntu ISO⁸ or similar, which automatically executes the following tools when introduced to the computer:

- **Hardware Discover:** obtains metadata from the computer and its components. The metadata includes different ways of identifying the device, and useful information for traceability, reusing, and for generating circularity inputs. It does not take any personal data, by default.
- **Eraser:** deletes private information. The deletion method is by rewriting different times the hard drive, as it is the only method to ensure deletion of data and leaving the hard-drive reusable.
- **Benchmark:** performs quality tests to guarantee the capacity of the machine to operate in the desired workplace correctly. This is interesting when compared with the usage of other computers and the results of their benchmarks, so we can predict the usage

⁷ “An agent can be a physical or virtual entity that can act, perceive its environment (in a partial way) and communicate with others, is autonomous and has skills to achieve its goals and tendencies. It is in a multi-agent system (MAS) that contains an environment, objects and agents (the agents being the only ones to act), relations between all the entities, a set of operations that can be performed by the entities and the changes of the universe in time and due to these actions” [61].

⁸ A live disk is a bootable OS, usually stored in an external media, as a CD, DVD or a pendrive. Ubuntu is a flavour (an adaptation) of Ubuntu for low-end computers, which uses lightweight versions of different programs.

(and potentially demand) of the computer. Benchmarks include CPU capabilities and hard-drive writing and reading speeds.

- **Diagnosis:** performs quality tests to ensure that the computer and its components work without failure. These tests include:
 - Hard-drives, by looking for broken parts, damaged sectors, other failed components and getting an estimated remaining lifetime.
 - RAM integrity and correctness of the sticks.
 - The overall capacity of the components and the computer for a real use, executing an up-to-date OS.
- **PXE server:** DeviceInventory is offered with a companion virtual image (a VirtualBox file) of a server, that can be used to register hundreds of computers at the same time using a local network (see Figure 7).
- **Filter:** automatically executes fast functionality tests and benchmarks, showing the final user a result encoded such as a traffic light: it emits a green light if the computer is fully operative and powerful enough to satisfy the demand, a yellow green if the computer is fully operative but there is no demand for it, and a red light if the computer is not fully operational. We can know about the demand thanks to the overall information of rejected and accepted devices in the DeviceHub of the preparer and in GRD.

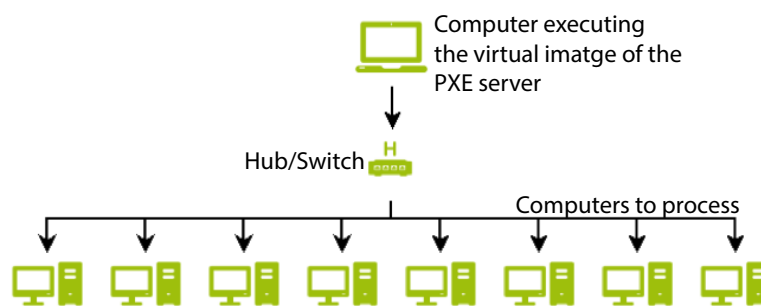


Figure 7 How the PXE server works.

The result of executing ‘Device Diagnostic’ and ‘Inventory’ is a non-modifiable, signed, and trusted report about the execution, which is uploaded to DeviceHub, as seen in Figure 10.

DDI is built on Python 3 and accepts Linux OS. The project repository is at <https://github.com/eReuse/device-inventory>.

5.1.2 DeviceHub

DeviceHub is an ITAMS focused on efficiently managing the circular life cycle of devices (see Figure 8 and Figure 9). It allows organizations to manage their hardware more effectively, avoiding unnecessary asset purchases (reduce), promoting the harvesting of existing resources (internal reuse), and interacting with external systems to manage the different processes involved in reusing and recycling, such as disposal, external reusing, tracking, etc. These systems can be other ITAMS (such as other DeviceHubs), other solutions, and traceability systems (such as the traceability system of eReuse.org, GRD). The following are the main characteristics of DeviceHub:

- Management of devices: manages the asset identification process and the life span of the devices of the organization, the workflow, and external ones over which the organization has control.
- Reports: generates automatic personalized reports and analytics, being able to ensure compliance, detecting critical aspects, such as loss of devices, over assignment of resources, etc.
- Certifications: saves, manages, and exports certificates, such as the results of the tests and deletion of the hard-drives, being able to answer to audits and show it to end users.
- Labelling: automatically generates personalized labels to tag the devices.
- Logistics, traceability, and tracking: Manages the placement and transportation of devices by mapping their coordinates to places and assigning responsible transporters and destinations.
- Packaging: manages the process of packaging multiple devices into boxes, and working directly with them.
- Advanced user managing: sets devices to users, whole departments, or custom hierarchical structures, ensuring the compliance of the organization policies over possessors, managers, employees and externals.
- Plug-ins: DeviceHub is built on an open-source framework, offering many plug-ins and is thought to be extended to very specific needs.
- Integration: DeviceHub, thanks to its nature, is thought to be integrated with other systems the organization already possesses.

DeviceHub is offered as a *web app*, split into two technical subprojects: a server, the DeviceHub project itself, and a browser client, DeviceHubClient. The server is a RESTful Python web framework built on Eve, a framework built on Flask, and uses MongoDB as the main storage engine. DeviceHubClient is a JavaScript, CSS, and HTML framework built on Angular, which offers an interface for users to interact with the API exposed by DeviceHub. The project repository is at <https://github.com/eReuse/DeviceHub> and at <https://github.com/eReuse/DeviceHubClient>. Section 6 details DeviceHub.

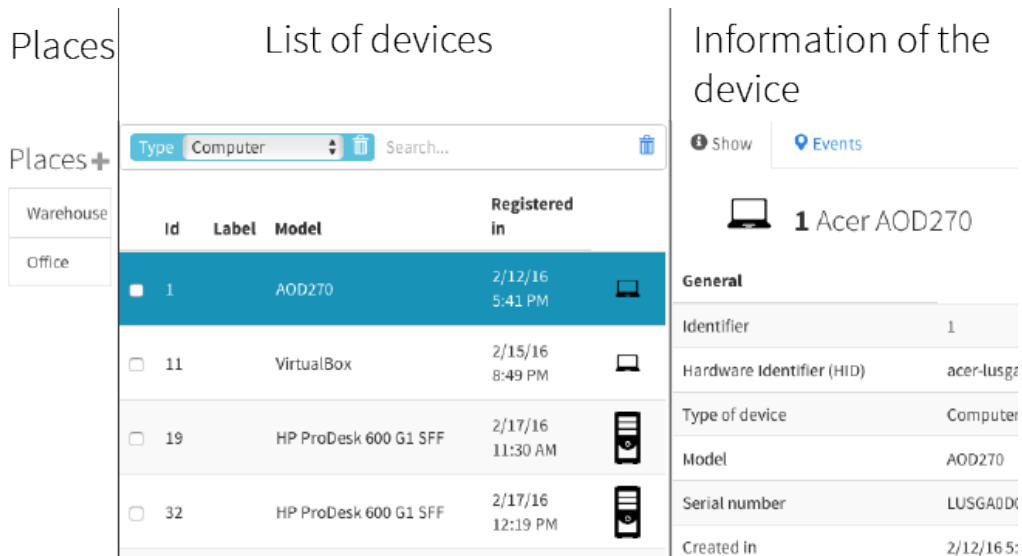


Figure 8 Main page of DeviceHubClient. The design is divided into three rows: a list of places, a list of devices and information for the selected device.

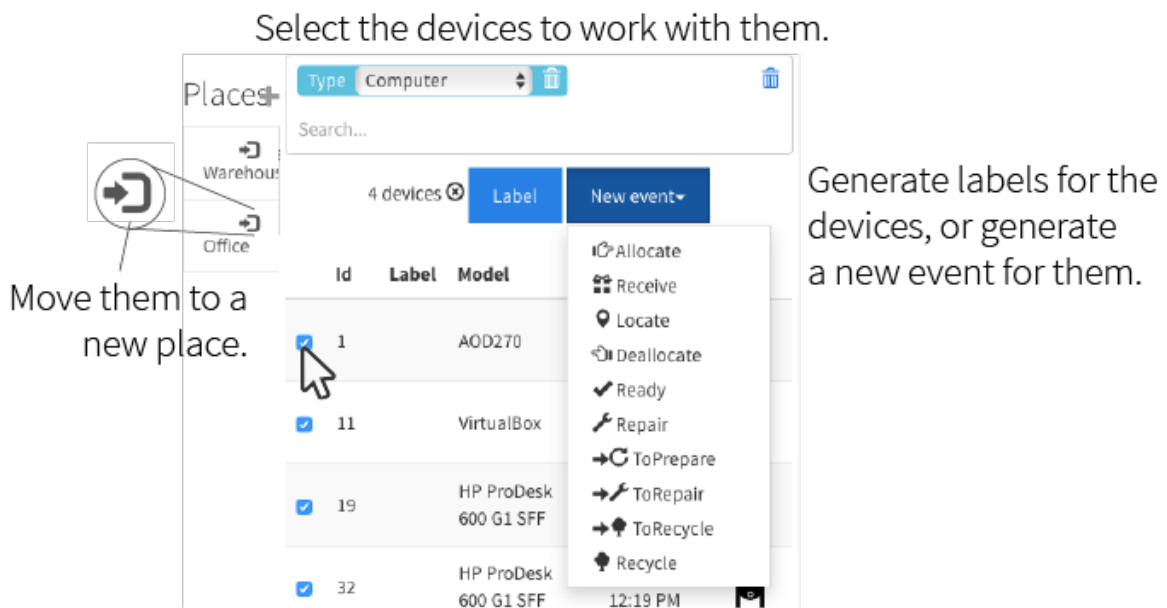


Figure 9 Users can perform different actions to multiple devices at once, such as move them to places, generate tags, or perform events such as repairing devices or allocating them to a client.

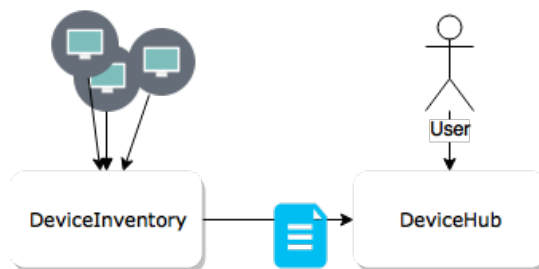


Figure 10 DDI automatically obtains metadata about a computer and its components. This information is then uploaded to a DeviceHub. Users interact with DeviceHub to manage reusing and disposal until collection.

5.1.3 Global Record of Devices (GRD)

The Global Record of Devices (GRD) is an online log that maintains a global list of traceability information about devices, so it can assist when there is leakage or similar problems. The GRD works with open-data and is designed to be used with external users who want to report traceability information. At the same time, GRD aggregates traceability data with the objective of using this data as inputs for indicators measuring circularity indicators.

GRD provides a REST API to allow ITAMS, as DeviceHub, to report the life cycle of devices, environmental responsibilities for organizations, etc. It collects from each device, at least, the geographical paths it has followed (not in detail to preserve privacy), the DeviceHub where it has been stored, the collection points where it was located before its final recycling and, in the case of computers, the record of its components. GRD provides a RESTful API, and is built on Django and MariaDB. The project repository is at <https://github.com/eReuse/grd>.

5.1.4 TransferHub

TransferHub is a system that manages the distribution channel, focused in simplifying transfers in the reusing process. It announces receivers and givers, so they can find each other easily. Optionally, it can add regulation or transaction mechanisms. TransferHub does not manage the donation itself: it interacts with DeviceHub or similar ITAMS to acknowledge transferences and to retrieve the available devices. TransferHub has three main functionalities:

- P2P: enables peer-to-peer transferences, without any regulation scheme. For example, P2P can be from many givers to many receivers, using the TransferHub just as a way to agree and announce the transferal, or from one giver to many receivers, in a more standard way, if it is a shop.
- Platform: adds rules and control mechanisms to the transferences, usually backed up by a community. These mechanisms include ways to control the reputation of the users, and ways to find the best match between receivers wanting devices and givers. A social platform is an umbrella that joins specific DeviceHubs from donors, receivers, and other platforms building an electronic reuse ecosystem. The main features are capturing offers (donations) and demand (social projects) to trade with reuse professional service providers (preparation, installation, maintenance), a donation service (legal advice), a payment system, a social support crowdfunding and a recommendation system. As an example, Reutilitza.cat uses a prototype of TransferHub with social platform functionality.
- Selling services: adds the necessary mechanisms to get paid by the transferal, as for covering costs or by seeking benefits from it.

TransferHub is built with Drupal 8, an open-source Content Management System (CMS) with a large community developing modules extending the core functionality, so it can be extended to fit any particular purpose.

5.1.5 eReuse.org App

The eReuse.org App is a smartphone and tablet client for DeviceHub. It is focused in performing certain actions – such as geolocating or visualizing devices- that benefit from the portability, camera and geolocation features of smartphones and tablets; and in registering them to DeviceHub by using a subset of functionalities from DDI. The main functionalities of the app are:

- Registering new geolocated places into DeviceHub and assigning devices to them.
- Quickly performing different actions on devices, avoiding the hassle of accessing a computer or manually entering data. Some of these actions need geolocation data, as they register where they have been performed. Examples, as in Figure 11, are receptions and disposals, where this data is used for efficiency and as a security check, avoiding mistakes, or bad intentions when entering the locations manually.
- Registering the smartphone or tablet to DeviceHub, obtaining traceability metadata and identifiers, and optionally performing a *factory reset*⁹.
- Assisting in registering devices other than computers by reading from QR or barcodes from already printed labels, avoiding manually entering the data.
- Obtaining information about a device by scanning its printed QR code. The code is the URI of the device in the DeviceHub where it has been registered.

The app is actually offered for smartphones and tablets running Android¹⁰ 2.3.3 or above, so it can be used by around 99% of the actual Android market [33]. This way, we ensure that old Android devices can be registered on the platform and can be reused (if they are too old to perform other tasks) for this purpose. The project repository is at <https://github.com/eReuse/eReuseAndroidApp>.

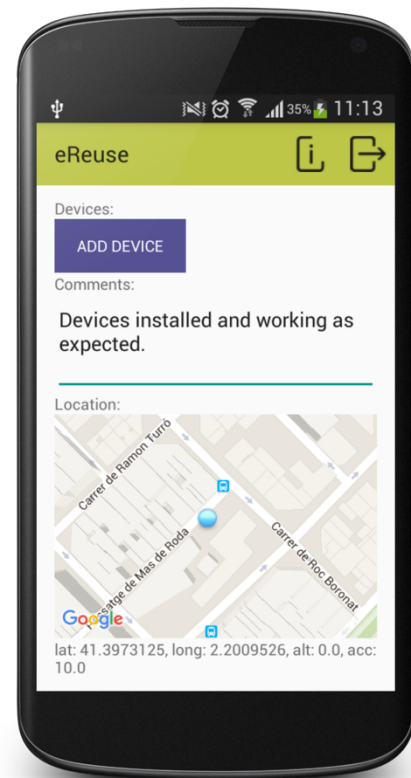


Figure 11 Representation of the interface a user sees in the DeviceHub App, when it is performing an action (what we call an “Event”) to a device. We can appreciate how the app shows the position of the user in a map. The position is going to be sent to the server, and it is going to be assigned to the device.

⁹ It means resetting the phone to factory defaults by erasing any personal data such installed applications.

¹⁰ At the moment of this writing, the app is offered on Google Play as *eReuse.org* at <https://play.google.com/store/apps/details?id=org.ereuse.scanner>.

Future work is to release an iOS (iPhone, iPad...), and a Windows 10 Mobile version of the app, so we can support the majority of the market. We might also support Symbian and Blackberry OS, if we find them as a suitable target for tracing old phones and smartphones.

5.1.6 Directory of Collection Points (DCP)

The Directory of Collection Points (DCP) shares collection places between different agents. For example, different DeviceHubs can reference to well-known places, which are stored in DCP (see Figure 12). The project repository is at <https://github.com/eReuse/dcp>.

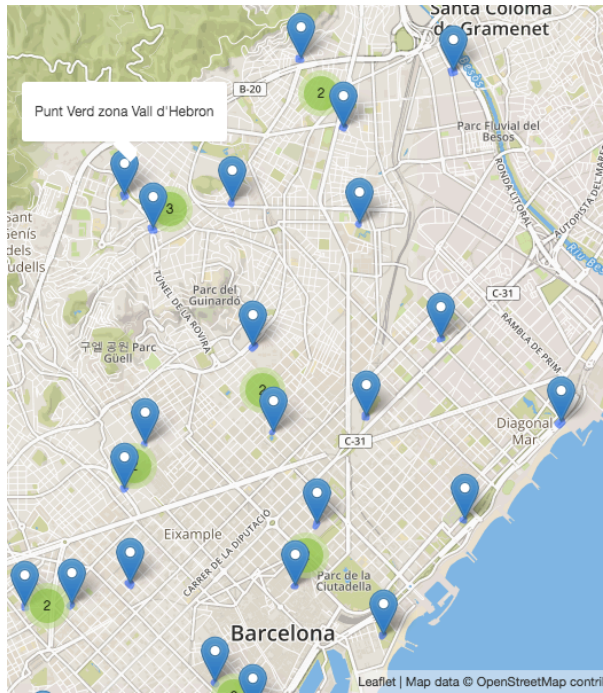


Figure 12 Map of collection points of Barcelona, as seen for DCP. The numbers on screen group collection points that are close.

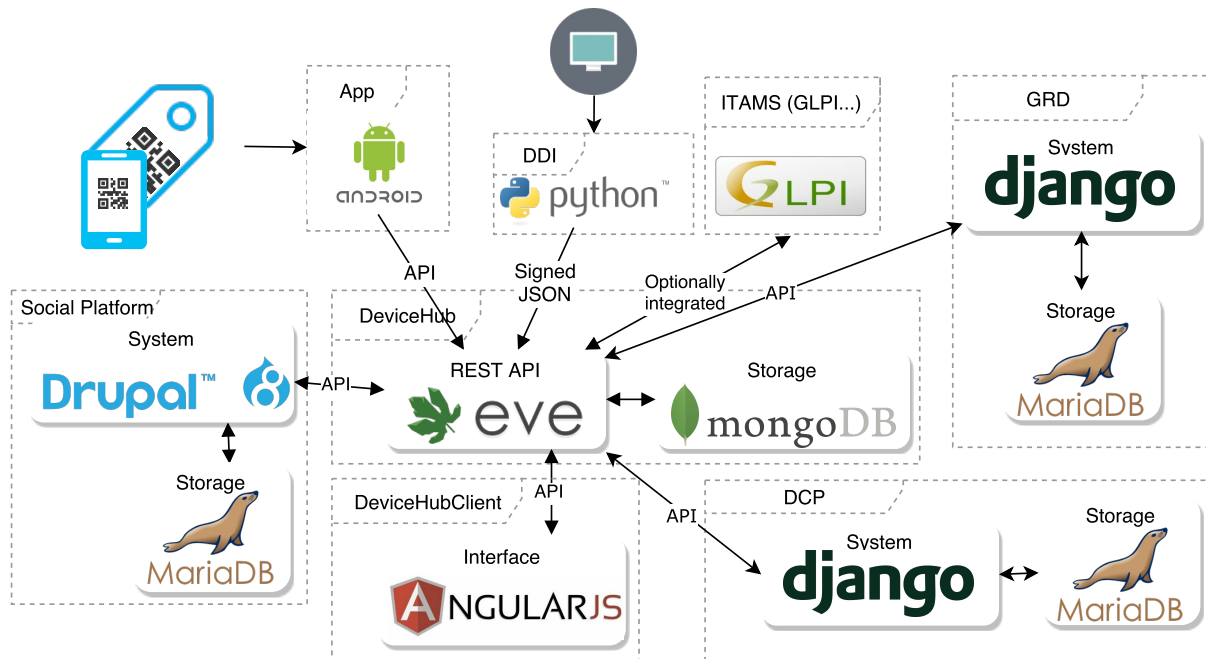
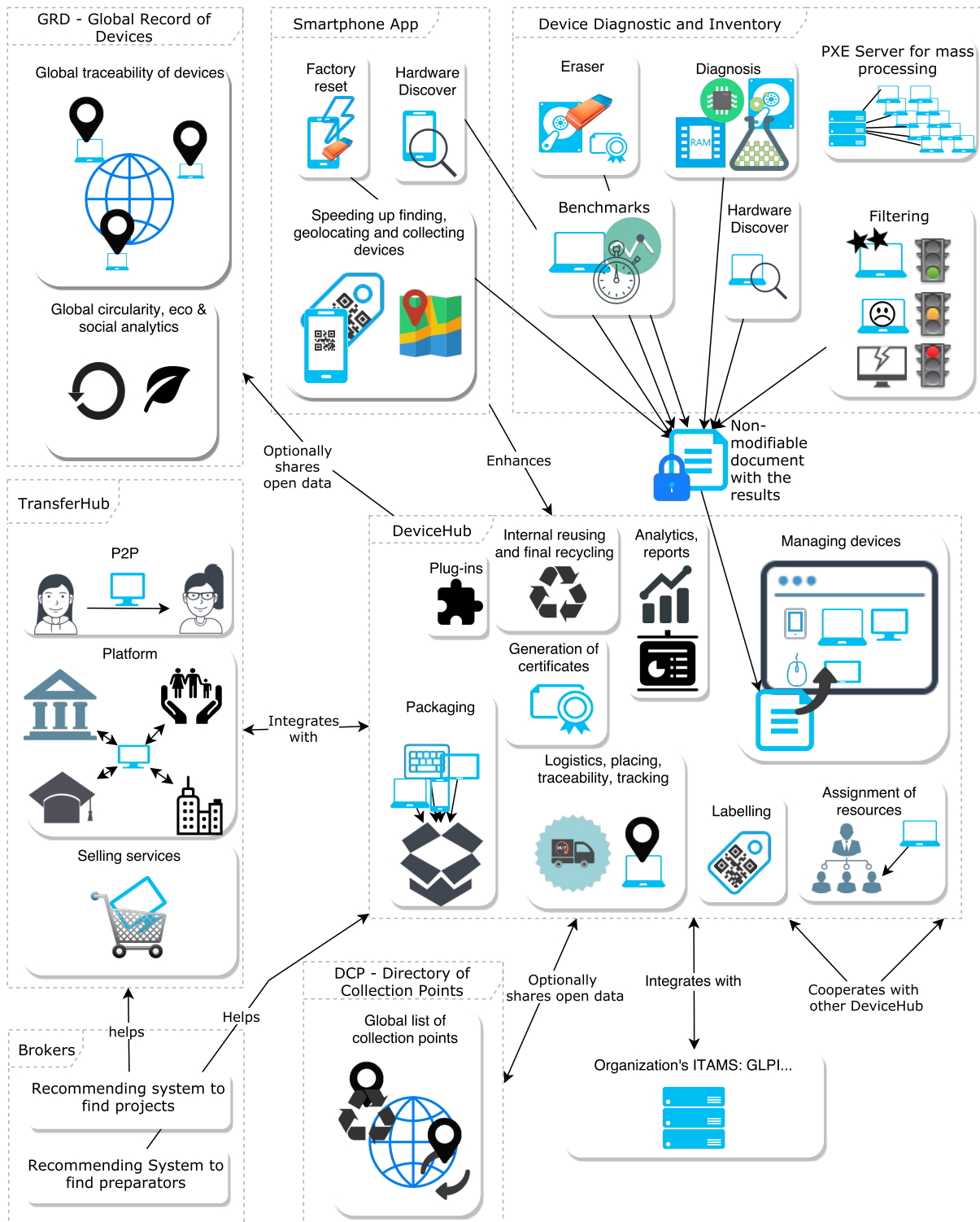


Figure 13 Technologies used by the components of the eReuse.org system and their relationships.



5.2 REPRESENTATION OF THE KNOWLEDGE AND API

The eReuse.org architecture has a common public schema that is used by all the agents, with slight differences between some of them. In this section, we justify and explain the ontology and the API.

5.2.1 Requisites for the exchange of knowledge

Collaboration between systems and organizations is at the heart of eReuse.org. To trace devices, there is a need for a global effort: stakeholders have to exchange information with eReuse.org (in concrete GRD and DCP) and use the provided tools to manage reusing, or integrate their own. The eReuse.org ontology is specially designed for system collaboration to succeed, based on being:

- Easy to understand and implement: so organizations do not need to do great investment when integrating the technology, auditors and managers can analyse the data in any moment, and third parties can analyse circularity indicators.
- Public: accessible for third parties.
- Versatile: the many types of devices and use cases we can encounter forces the ontology to be versatile enough to adapt to them, by being easily extended and modified. This is most likely going to happen when new types of users adopt the system, who usually have new use cases or special needs.

5.2.2 Definition of the Ontology¹¹

In this section, we are going to describe the ontology of eReuse.org. Note that agents generally use a subset of the ontology, and GRD has some small differences regarding the schema of few fields. DeviceHub adheres to this definition (see Section 6.3 for specifics).

5.2.2.1 Usage of existing ontologies

At the moment of writing this thesis, as there is no available specific ontology (regarding devices) from which we can extend; we do it from Schema.org, embedding other ones when needed. Schema.org is “a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet” [34], and specifically, we use the official OWL version of Schema.org from Holger Knublauch [35]. Schema.org’s main sponsors are Google, Microsoft, Yahoo and Yandex, and is backed up by the W3C and the community. It was born as a way to show better search results, a common ontology for search engines to understand websites, and it has been widespread through the Internet: in 2014 22% websites from a sample of 10 billion had schema.org mark-up, and in 2015 were already 31.3% [36]. Communities in the W3C are already working with organizations to extend Schema.org in selected domains, as the Automotive Ontology Community Group [37], which is focused in the automotive industry, and the bib.schema.org, focused in the bibliographic sector [38]. It is in the top 10 vocabularies in LOV [39], and it is gaining popularity.

We have chosen Schema.org (instead of *dcterms* or SKOS for example) mainly because it is easy to understand and adopt, thanks to its easy-to-use and explicit documentation [40] that benefits newcomers; it accepts well known formats, such as Microdata, RDFa, JSON-LD, and OWL; it is widely adopted (although not as a way to consume API as yet), so it can be easy for

¹¹ “An ontology is a formal, explicit specification of a shared conceptualization.” (Studer, Benjamins, & Fensel, 1998)

them to use a language that extend it; and our agents can benefit from SEO strategies without transforming any data.

Apart from Schema.org, we use other vocabularies when this one is not enough, and we link when possible to geonames.org and to Wikipedia through The Product Types Ontology [41].

5.2.2.2 Conventions

We present an undisclosed list of conventions when defining a new type of resource:

- For a given resource, we first try to use Schema.org when possible. Otherwise, we create a new resource, extending Schema.org.
- New resources use the name of The Product Types Ontology when it makes sense to, and they link to it.
- We replace geo-coordinates and GeoShape from Schema.org to their equivalents from GeoJSON, as there is native support for the latter in many frameworks, including the ones we use.

5.2.2.3 The vocabulary

The vocabulary of eReuse.org can almost be represented by the classes of device, event, account, and place, and their subclasses. In the following sections, we will explain them as they are in DeviceHub (the GRD and DCP differ in some fields). Note that other systems may have a subset of this vocabulary.

5.2.2.3.1 Devices

Device, as explained in Section 2.1.2, is everything potentially reusable, and is represented in Figure 15. Some devices can have inner devices called components (see Figure 16). The relationship is recursive: as components are devices that can have inner components, too. For example, both a computer and a graphic card are devices, and the transistors of a graphic card are devices. Although designed as an infinite recursive, the agents in eReuse.org only support one layer of recursivity: this is `Computer` → `GraphicCard`, but not `GraphicCard` → transistor.

`Device` extends `IndividualProduct` (from Schema.org). The result is a class describing physical characteristics from a device (width, weight...), identifiers (serial number, HID, synthetic identifier, URL...), and other metadata (icon, manufacturer name...). `Device` is extended by:

- `ComputerMonitor`¹²: includes any display such as TFT or LCD screens. Although actual recorded information is the inches, future work is adding resolution, colour scheme, and other specific related information.
- `Mobile` (no equivalent in product ontology): smartphones and tablets, for which we can record the IMEI and MEID. `Mobile` is extended by `MobilePhone`¹³ and `TabletComputer`¹⁴.

¹² Product ontology in: http://www.productontology.org/id/Computer_monitor

¹³ http://www.productontology.org/id/Mobile_phone

¹⁴ http://www.productontology.org/id/Tablet_computer

- Computer¹⁵: machines that can run the Device Diagnostic and Inventory software, such as Desktop, Laptop, Netbook, Server and Microtower.
- Peripheral¹⁶: a wildcard for any “device that is used to put information into or get information out of the computer” [42], which does not fit in any other subclass and does not have specific characteristics that should be recorded in an own specialized subclass. Actual types of peripherals accepted are Router, Switch, Printer, Scanner, Multifunction printer, Terminal, HUB, SAI, Keyboard and Mouse.
- Component¹⁷ (from Electronic Component): devices that are part of other devices. Some components can have benchmarks; see Figure 20 for their representation. We accept the following components:
 - GraphicCard¹⁸. Future work is to incorporate, or to link to existing benchmarks about the model of graphic card.
 - HardDrive¹⁹. We collect different tests and benchmarks.
 - Motherboard²⁰. We collect the types and number of connectors the motherboard has.
 - NetworkAdapter²¹ (from Network Interface Controller), using the MAC Address as the identifier when creating the HID.
 - OpticalDrive²²: CD, DVD, HD-DVD and Blu-ray readers and recorders.
 - Processor²³ (from Computer Processor): as an abbreviation of Central Process Unit (CPU).
 - RamModule²⁴ (from Random Access Memory).
 - SoundCard²⁵.

¹⁵ <http://www.productontology.org/id/Computer>

¹⁶ <http://www.productontology.org/id/Peripheral>

¹⁷ https://www.productontology.org/id/Electronic_component

¹⁸ http://www.productontology.org/id/Graphic_Card

¹⁹ http://www.productontology.org/id/Hard_drive

²⁰ <http://www.productontology.org/id/Motherboard>

²¹ http://www.productontology.org/id/Network_interface_controller

²² http://www.productontology.org/id/optical_drive

²³ http://www.productontology.org/id/Computer_processor

²⁴ http://www.productontology.org/id/Random-access_memory

²⁵ http://www.productontology.org/id/Sound_card

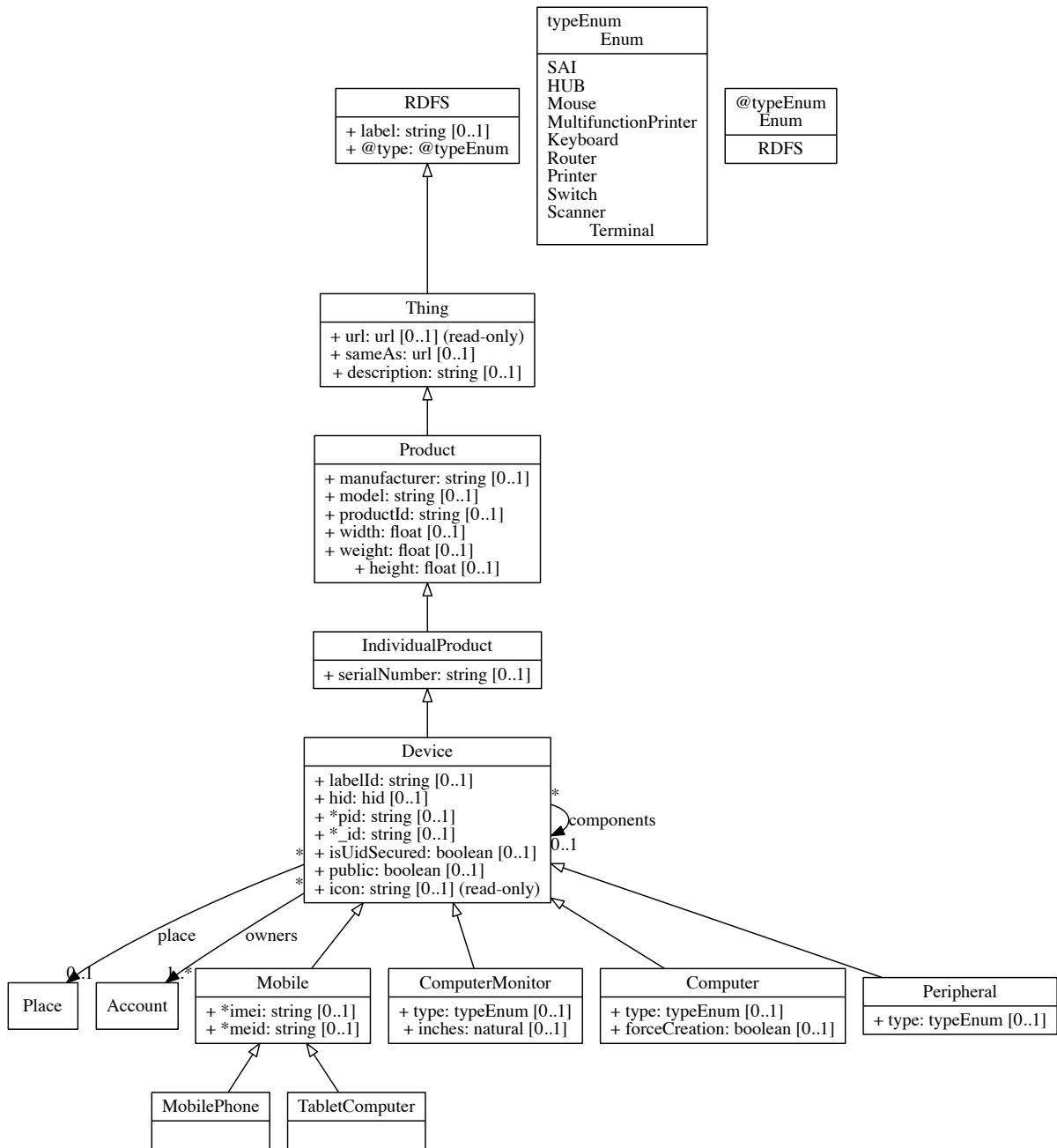


Figure 15 Class diagram for Device and types of devices, except Component, which is shown in Figure 16.

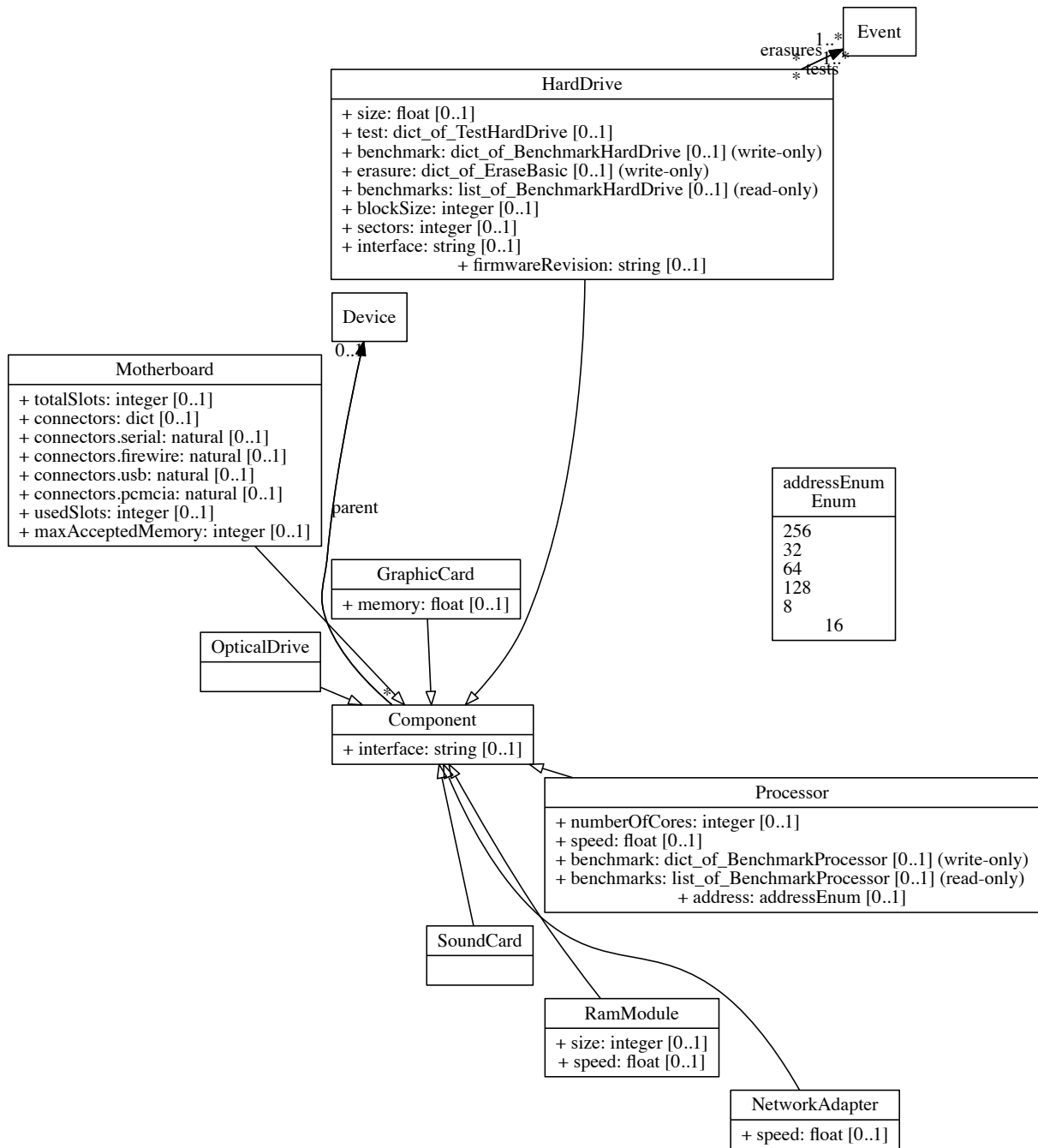


Figure 16 Class diagrams for Component and its subclasses. Component extends Device, as shown in Figure 15.

5.2.2.3.2 Events

Events are the actions performed to resources such as devices. For example, to say to the system that a device has been repaired, we will perform the event `Repair` with the concrete device as a parameter. Event extends `Event` from Schema.org, with attributes defining where it happened (by defining a place or by geo-coordinates), who performed it, when (both user defined date and system dates), and a control if the event can be considered secured (because it has been checked by the system or it has been automatic), etcetera. DeviceHub and GRD store, such as in a log, the events performed to a device, successfully monitoring its life cycle (see Figure 17).

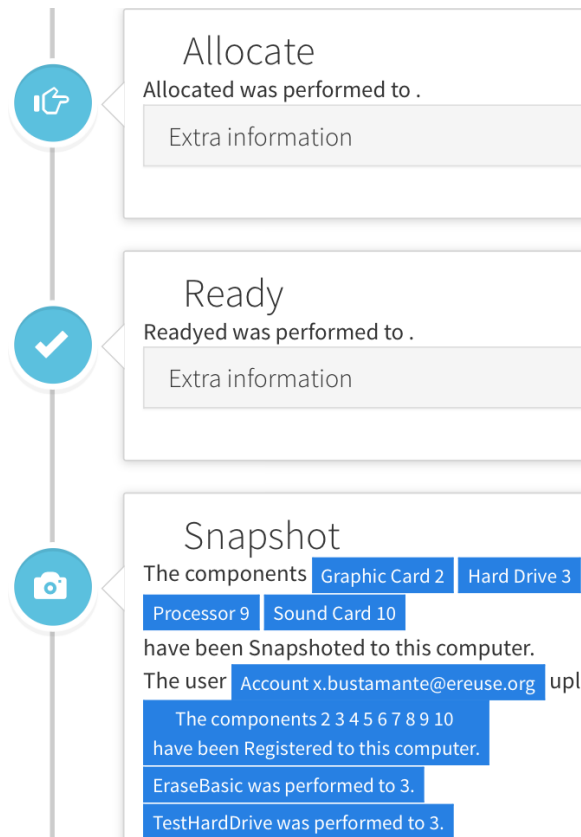


Figure 17 List of events for a device in a DeviceHub, in ascending chronological order (bottom is first). For this list, we know that the device has some components, and the hard-drive was tested and erased (see below the event Snapshot). After this, the device was set to be Ready to be reused, and finally it was allocated to someone.

Event is an *abstract* class supposed to be extended to cover any specific need. This project explains the events regarding devices (Table 4, Figure 18 and Figure 19); however, at the moment of this writing, there are starting to draft proposals for events for social projects and sales (for TransferHub).

Classes extending events need to be written following the general conventions (PascalName), and they represent a verb in the infinitive when possible. Some events represent the *willingness* or *assignment* to do an action (toAllocate vs Allocate, toPrepare vs Prepare, toDispose vs Recycle...). These verbs may have the preposition *to* as a prefix, and written in camelCase: Allocate → toAllocate.

Table 4 Table describing the events regarding devices with the systems that use them. To understand the events fully, we recommend reading the rest of the eReuse.org architecture, where they are presented within their context.

Name	Description	DeviceHub	GRD	App	DDI
Accept	The user or organization accepts the toAllocate done to it. After this, the devices are assigned to it.	✓			

Add	A component is added to a device.	✓	✓		
Allocate	The device has been assigned to a user or an organization. The allocated users or organizations are responsible for the device.	✓	✓		
Deallocate	The reverse of allocate. Removes the assignment from a user or an organization.	✓	✓		
Dispose	The device has been correctly disposed.	✓	✓		
EraseBasic	The hard drive has been erased in a fast way. A certificate can be generated from this event.	✓			
Free	A device is freed (made available) when there is willingness for it to be donated or used, and the device works correctly.	✓			
Locate	The device has been located.	✓	✓		
Migrate	Changes the holder agent of the device (see Section 5.2.3). Migrate is a final state for a device in an agent. The events after migration need to come from the new agent.	✓	✓		
Ready	A device is ready when it has been assured that it works correctly.	✓			
Receive	The receiver, a user or an organization, confirms that the device has arrived. There are the following types of reception: <code>RecyclingPoint</code> , <code>CollectionPoint</code> and <code>FinalUser</code> .	✓		✓	
Recycle	The device has been recycled. This is the end of its lifetime.	✓	✓		
Register	The device has been registered (created) on the system. In general, this is the start of its lifetime (from the point of view of the eReuse.org software).	✓	✓	✓	
Reject	A user or an organization refuses a <code>toAllocate</code> done to it.	✓			
Remove	A component has been removed from a device.	✓	✓		
Repair	A device has been repaired.	✓			
Sign	In some scenarios, users need to sign a document specifying their acknowledgement of the agreements and conditions when receiving devices. This is represented by this event, specifying the URL of the signed document and the affected devices.	✓			
Snapshot	Updates the ITAMS so the state and events of its devices are the same as represented in the parameters of the <code>Snapshot</code> (see Section 6.2.3).	✓			✓
TestHardDrive	A test has been performed to a hard drive. The tests check for the integrity of the hard drive. DeviceHub can generate a certificate from the data of this event.	✓			
toAllocate	Tries to allocate a device to a user in an organization. After a <code>toAllocate</code> is performed: (1) user must accept	✓			

	it or reject it, and (2) if user accepted it, the system will perform <code>Allocate</code> .				
toPrepare	A device has been selected to be prepared. Usually is the next event done after being registered.	✓			
toDispose	The device must be disposed. It does not say to which collection point the device is going to be taken to, we can extrapolate this from <code>Allocate</code> .	✓			
toRepair	A device has been selected to be repaired. This event will probably require a message for the technician to know what to repair.	✓			
UsageProof	The device is on use. This event can be sent periodically.	✓	✓		

Allocate, Deallocate and Receive usually present some confusion, so we explain the differences: `Allocate` assigns the user or organization that has some kind of property over the device. `Allocate` can be performed on different users, and all of them will share the property. `Deallocate` removes the property from one user. On the other hand, `Receive` sets the device physically with the user or organization. Performing `Receive` again to another user will move the device to that one. The system is quite granular, and it is up to the organization to adopt a more or less rigorous way to apply the events.

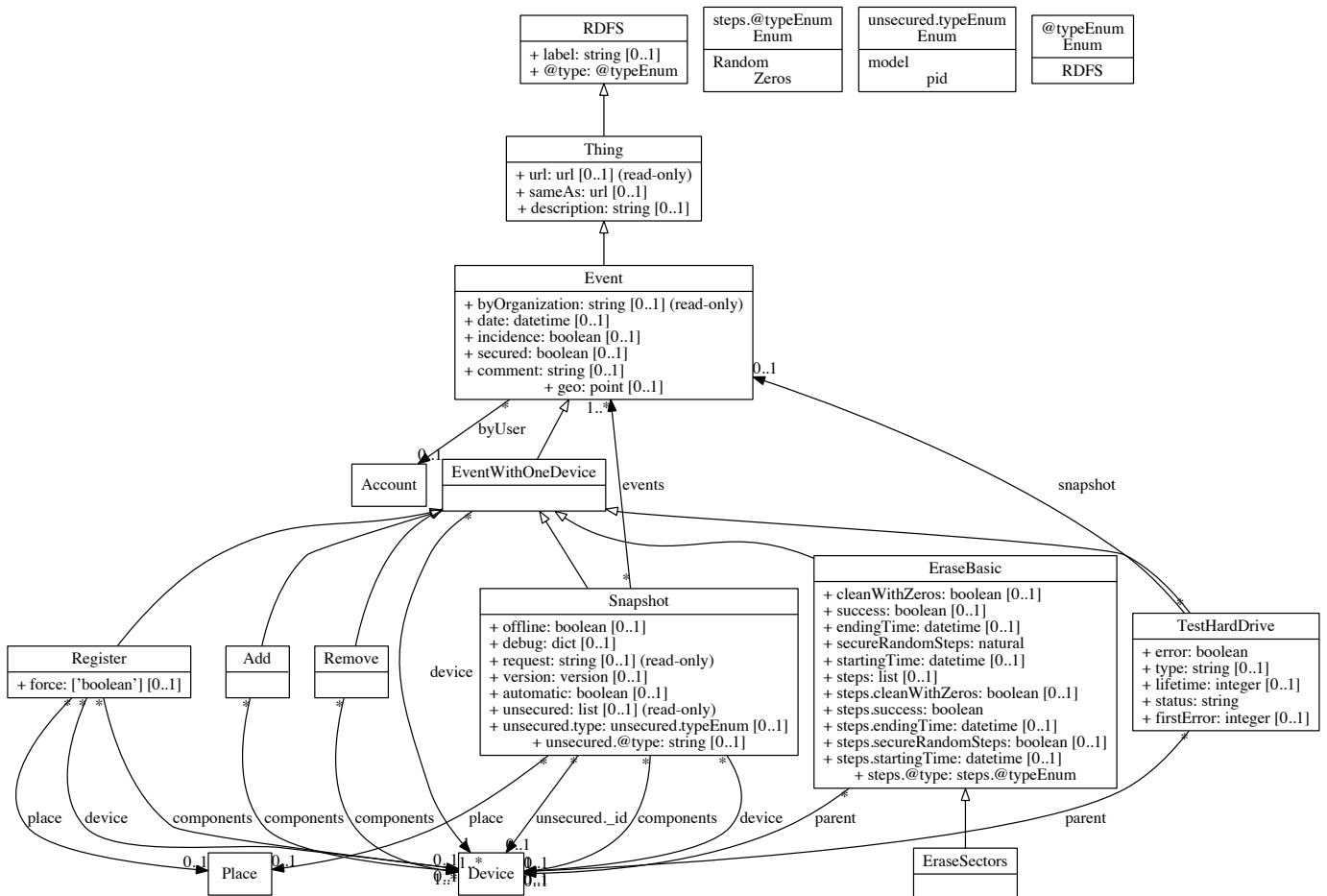


Figure 18 DeviceHub Class diagram for Event, and events with one device.

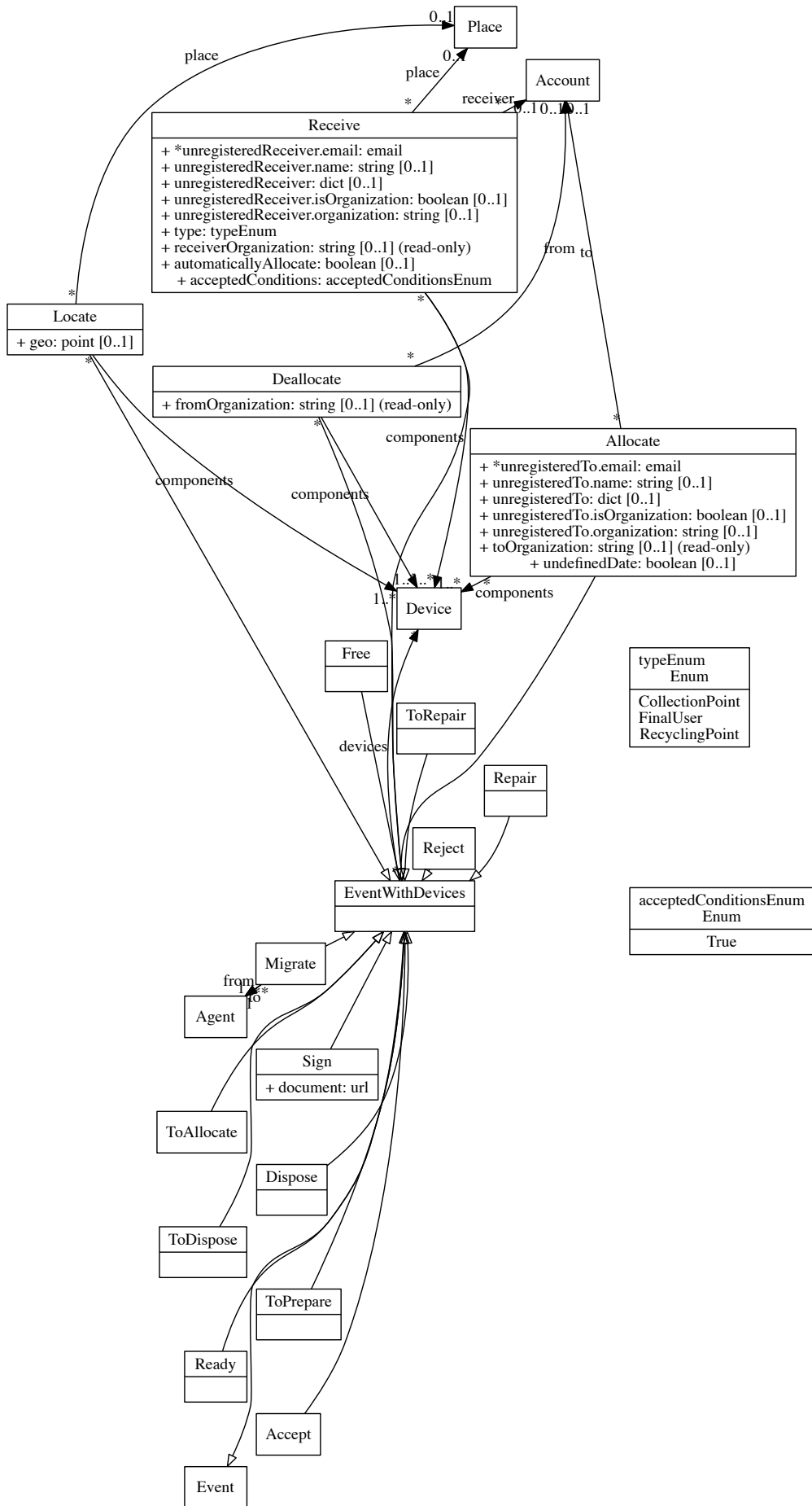


Figure 19 DeviceHub class diagram for events with multiple devices.

5.2.2.3.3 Places and accounts

Places define interesting points in the life cycle of devices, where events occur and in which devices are located (see Figure 20). Places can be represented by an area, so any event happening inside such coordinates is tied to the place. Place represents the homonymous class in Schema.org.

5.2.2.3.4 Accounts

Accounts represent users or organizations (see Figure 20).

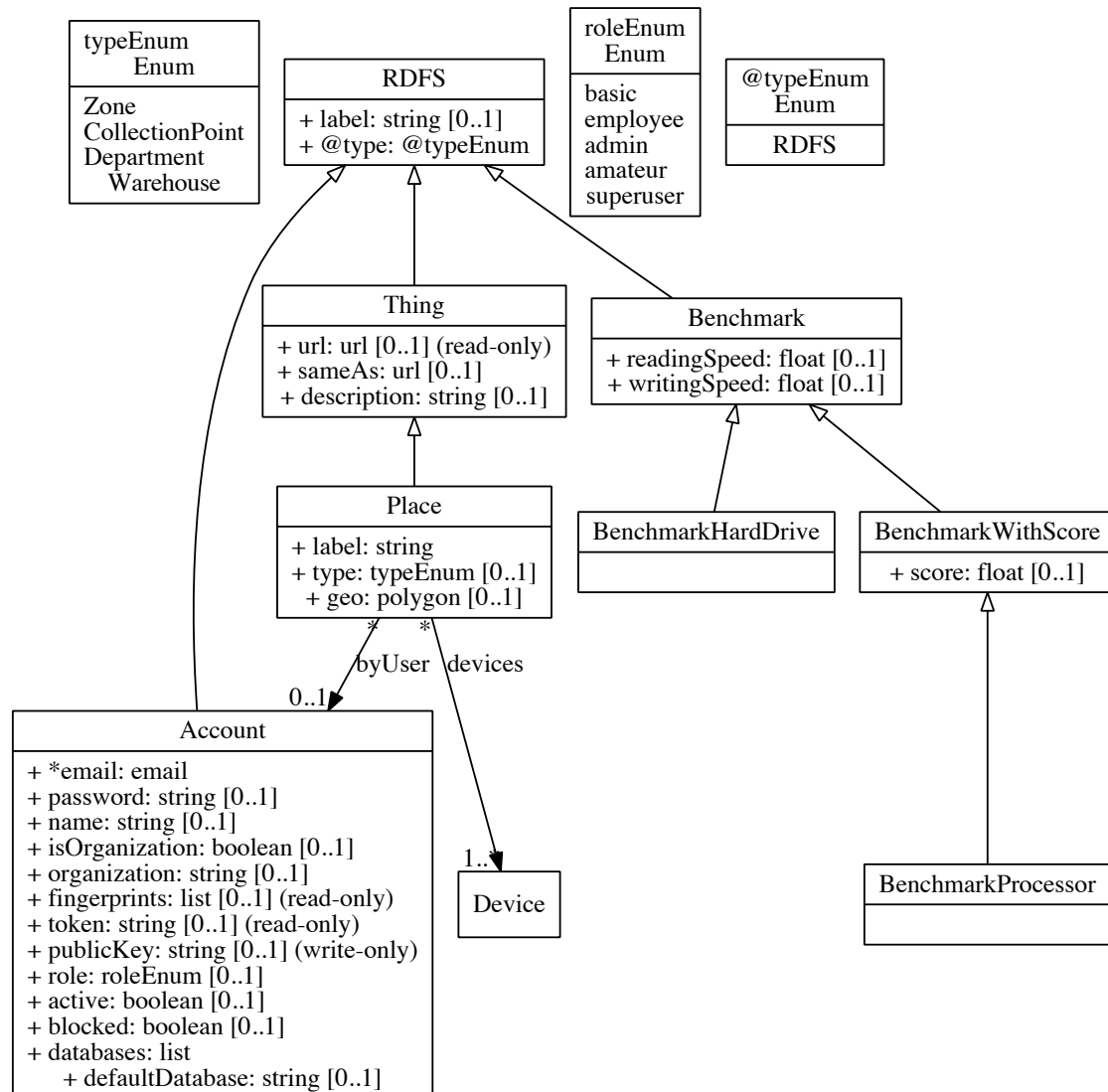


Figure 20 DeviceHub class diagrams representing Place, Account and Benchmark.

5.2.3 The URI as the main identifier

The environment of the eReuse.org system is the Internet, and there are many and different agents (some made by eReuse.org, some made by others) that interact with devices, events, etc. For those reasons, we use the URI for the main identifier of any resource, as it is the proposed universal identifier by the WHATWG [43] and is, definitely, the basis of the Internet.

In our architecture, we define the concept of *main URI*, which is the URI from the ITAMS that *holds* the resource. Given a resource such as a device, its holder is the only agent with the right to perform events. For a device, at least one user or organization inside the agent needs to possess it physically and legally. For devices, the *main URI* changes in their life cycle as they travel through different ITAMS. We use `sameAs` as an ordered log for the URI (see an example in Figure 21). Therefore, agents link resources keeping references in the field `sameAs`. The protocol is http or https.

















Time 1	Time 2	Time 3	Time 4
  Agent A			
	  Agent B		  Agent B
 Agent C	 Agent C	  Agent C	 Agent C
Device	Device	Device	Device
+ url: Agent A + sameAs: []	+ url: Agent B + sameAs: [Agent A]	+ url: Agent B + sameAs: [Agent A, Agent B]	+ url: Agent B + sameAs: [Agent A, Agent B, Agent C]

Figure 21 An example of how the field `url` and `sameAs` work.

The URI is decomposed as follows:

```
https://api.devicetag.io/reutilitza-cat/devices/acer-321-iconia
```

Protocol	Host	Database	Resource type	Resource identifier
https	api.devicetag.io	reutilitza-cat	devices	acer-321-iconia

- Protocol: If not added, it is supposed to be http or https.
- Host: As defined for the W3C: a domain, IPV4, or IPV6. We recommend, for the sake of readability and simplicity, a domain.
- Database (optional): agents such as DeviceHub can be split into different databases representing, for example, different organizations. By specifying this into the URI, we can know the organization owning a device by only reading the URI, avoiding repeating identifiers. See Section 6.2.1 for more information.
- Resource-type: The name of the resource.
- Resource identifier: A text uniquely identifying the resource inside the agent or inside the database. Some organizations have an own-defined identifier (named synthetic

identifier), but we recommend, when possible, the HID. See Section 5.4.1 for more details.

5.2.4 API

Although the system in eReuse.org is heterogeneous, there is a common definition for the API of every agent. In general, the API is RESTful serialized with JSON-LD. In the following paragraphs, we explain the common concepts and justify the decisions.

5.2.4.1 Technologies used

We have three main methods to represent the above-described knowledge using an access point: SOAP, REST, or a SPARQL point.

SOAP is the W3C standard for the exchange of data between agents on the Internet. Designers specify the operations of the service (in WSDL and in other documentation) and clients need to adapt to them. SPARQL is the standard query language for RDF, being similar to SQL. Clients can build a personalized query and use it against your servers. Representational State Transfer (REST) is a set of architectural constraints focused on hypermedia, on linked resources [44]. Proposing standard constraints narrows diversity but focuses decoupling and reduces overhead in implementing clients for services.

While widely used, SOAP creates a tightly coupled system that requires an overhead for the client to understand the context of the service; this goes against our objectives. SPARQL provides a highly personalized way of consuming data, which may be interesting for analytical purposes; however, it lacks the operational part. REST proposes standard constraints that make it incompatible with some servers; however, it also decouples our systems with the clients and offers a well-known set of mechanisms, which follows our objectives. eReuse.org systems are based on resources (devices, events, places and accounts) that we identify with URI, which is the basis of REST.

For the data exchange format, we use JSON over XML as it is simpler, more human readable than XML [45], and it is already used by some of our systems as part of its code, reducing impedance mismatch. As opposed by RDFa, JSON does not have a standardized way of representing knowledge; this is why we use JSON-LD, which is natively compatible with Schema.org, and can be translated to other languages as RDF. As W3C, “JSON-LD is a useful data serialization and messaging format. [...] It is primarily intended to be a way to use Linked Data in Web-based programming environments, to build interoperable Web services, and to store Linked Data in JSON-based storage engines.” [46].

Finally, we use a small subset of Hydra, a draft from the W3C that “is a lightweight vocabulary to create hypermedia-driven Web APIs” [47]. For future work, it should be used more in the measure in which we need it.

5.2.4.2 Definition of the common API

In eReuse.org, resource types are represented by `@type` and `@context` is defined in the link header, as seen below:

Listing 1 Example of an HTTP link header representing a Device. Although in the specification, is future work to create the ontology files.

```
<http://www.ereuse.org/ontology/device.jsonld>; rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"
```

5.2.4.2.1 Errors

Errors extend from the *Error* class in Hydra and are represented as follows: the `@type` field describing the type of class or error, a `title` field representing a translated human title, and a `description` representing a translated human description of the error.

Listing 2 Example of a response containing the error `UnauthorizedToUseDatabase` from DeviceHub.

```
HTTP 401
<http://www.ereuse.org/onthology/UnauthorizedToUseDatabase.jsonld>;
rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"

{
  "@type": "UnauthorizedToUseDatabase",
  "title": "Unauthorized to use the database db1",
  "description": "User has no access to this database."
}
```

As you can see, the server sets the most specific HTTP status code.

5.3 MANAGING REUSING

In this section, we explain how the architecture manages the two main processes involved in reusing (which are presented in Section 2.2.1): the preparation for reuse and the transferal.

5.3.1 Preparation for the reuse

The preparation for reuse is the process in which the organization readies a device for reusing it. This process is usually done by a specialized department or organization, which are or act as preparers; it starts when they receive the devices and finishes when they are ready to be reused. The main objectives of the process are: (1) obtaining the necessary information of the device to trace it, (2) refurbishing it (ensuring it works correctly, etc.), and (3) ensuring it has potential demand.

Although there are different ways and steps to perform the preparation for reuse (specially its ordering) we show in Figure 22 one of the most used, based in our experiences, which is an extension of the recommended protocol of the Waste & Resources Action Programme (WRAP) [48].

The preparer can use DeviceHub to manage the process, updating the state of the machine through the system, annotating incidences or writing useful comments to colleagues. It will use Device Diagnostic and Inventory (DDI) in some key steps of the process, automating and securing tasks.

The preparer realizes, for all the input devices, the process as follows:

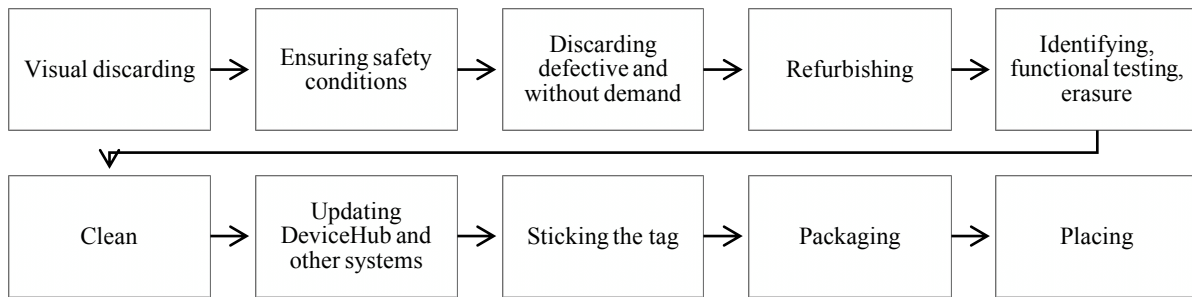


Figure 22 An exemplifying preparation for the reuse process, extended from WRAP [48]. Note that identifying can be done at the beginning of the process and can be repeated after refurbishing the devices.

It is interesting to note that the first three steps act as filters and their objective is double: first, to ensure that the devices are secured and operative, and second, to avoid investing resources in devices that are going to end up being directly recycled, not reused, as there is no demand for them.

Prior to starting the preparation for the reuse, some organizations annotate all the received devices, by performing `Receive` if the devices were already registered in an ITAMS (see Section 6.2.5.1 for an example of `Receive`), or by identifying the device, such as in the step *Identifying, functional testing, erasure* explained in the figure above. Some time may pass between reception and preparation, such as when preparers are waiting for a greater batch with which to work. In these cases, they can emit `toPrepare`, as a way to remember that those devices are waiting for preparation. Many devices may be found broken during the process. In these cases, preparers can use `toRepair`, to set that those devices need repairing, or `toDispose`, if the devices need to be disposed. Devices are considered to be repaired by issuing `Repair`. Finally, a device is successfully prepared for reuse when `Ready` is set.

5.3.1.1 Visual discarding

The preparer visually inspects the devices, detecting any fatal problems prior to running the software, such as broken parts, etc.

5.3.1.2 Ensuring safety conditions

The preparer performs a safety check, ensuring voltage and other safety requirements.

5.3.1.3 Discarding defective and without demand

The preparer performs some tests to ensure that the device works correctly, testing the components and the overall machine.

If the device is a computer, this process can be automatically done by the module Filter of Device Diagnostic and Inventory (DDI), as explained in Section 5.1.1. Briefly, this module performs some lightweight tests and benchmarks to know, in a glance, if the computer is operational and whether there is potential demand for it. After this last step, the preparer knows that it can invest in the device, as it works and there is potential benefit.

5.3.1.4 Refurbishing

In this step, the preparer repairs the device by changing components, and soldering or welding parts. Note that the components might have already been prepared for reuse, and DDI can track additions and removals of components from computers.

5.3.1.5 Identifying, generating metadata, testing, erasing

The preparer generates all the possible metadata for the device to identify it, performing the first step in tracing it, and optionally contributing data for the circularity inputs. In the same step, if programmed for, it will fully test the device (as an opposite from 5.3.1.3), and comply with privacy and protection laws by erasing any hard drive found.

This process can be done in different ways, depending of the type of device:

For computers, the preparer introduces a live CD with Device Diagnostic and Inventory (DDI), via a CD or USB. If there are numerous devices to prepare, it can execute DDI in batches by using the network with the PXE Server offered with DDI. For each computer, DDI can perform:

- Hardware Discover: obtaining the metadata of the device to identify it, etc.
- Eraser: deleting private information, complying with the standards. The result of this process is the events `EraseBasic` and `EraseSectors`.
- Functionality tests of the hard drive, RAM, capabilities of the device, etcetera, resulting in, for example, `TestHardDrive`.
- Generating a non-modifiable report, which is a JSON file containing the event `Snapshot` plus information about the signature.

For smartphones, tablets, or similar devices executing a mobile OS such as Android, the preparer can use the eReuse.org App, which executes a customized Hardware Discover, extracting the metadata of the device and automatically registering the information to a DeviceHub.

For devices containing labels with barcodes or QR codes, the preparer can use the eReuse.org app to read from them, avoiding introducing the serial number manually. This process still relies on the user introducing the rest of the information.

For the rest of devices, the preparer will need to introduce the information manually to its DeviceHub, via a web form. It requires at least one identifier, and we encourage introducing the information necessary to generate a HID. We explain it in Section 5.4.1.

5.3.1.6 Clean

Once the device is operational, refurbished, and registered, it is visually improved, so it is more appealing to the future receiver. This process usually changes depending on whether the device is for a donation or for selling, but it generally includes cleaning and painting.

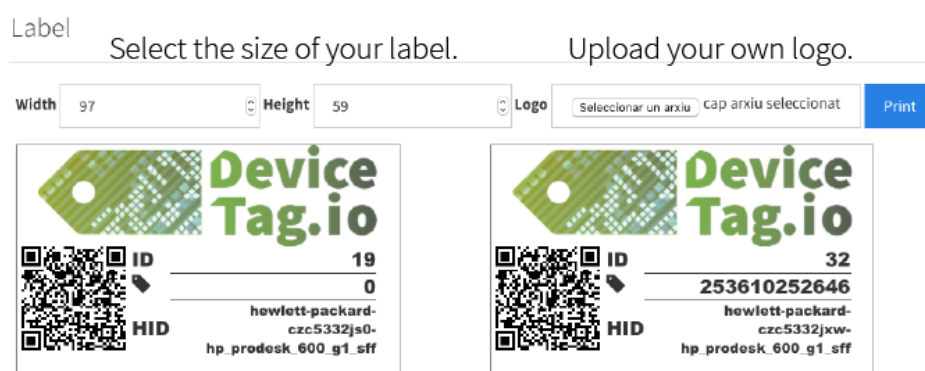
5.3.1.7 Updating DeviceHub and other systems

The preparer uploads all the reports, at once, to the DeviceHub of the organization. For smartphones, this process has already been done by the app automatically.

As explained above, the report is a file containing a JSON encoded `Snapshot`. DeviceHub updates its own database to reflect the changes described in the `Snapshot`, and will update other DeviceHubs if necessary. Finally, it will send metadata described in the `Snapshot` about traceability and circularity inputs to GRD. This process is detailed in Section 6.2.3.1.

5.3.1.8 Generating the tag

After registering the devices in DeviceHub, the preparer can print the tags for the devices. Tagging them is important to identify devices visually, for example when having to pick them up, or by using the eReuse.org App. In Figure 23, we can see a presentation of the web interface where the preparer can visualize the tags for every device before printing, and can personalize them: it can change the width, the height, and the logo of the tag. This process is detailed in Section 2.



See in real time your result, and generate a PDF by pressing “Print”.

Figure 23 The preparer can personalize the labels and preview them in real-time. Once it is satisfied, it can “print” them to a PDF. Thanks to the co-design aspect of the development, we acknowledge that the majority of our users prefer PDF to printing directly, as they can send it to the co-worker responsible for printing, and they can always print the PDF easily.

5.3.1.9 Packaging devices

It is normal to manage a group rather than individual devices. For example, a group containing a desktop computer, a keyboard, a mouse and a computer monitor (for a final user), or a group containing different desktops of the same model (for organizations that want the same model). In these cases, devices are packaged together, and the package can be registered in the system and tagged as if it were a device. Thanks to this, the preparer knows what is inside the package without the need to open it (reading the tag with the eReuse.org app, for example) and it can operate with it directly. In addition, events affecting a package are transmitted to its devices; for example, when a package is sold all the inner devices are sold, too.

Creating the package system is future work for DeviceHub.

5.3.1.10 Placing devices

Finally, the preparer introduces in its DeviceHub the location where it is leaving every device, waiting to be picked when transferred to its new owner—this way the transferal can be as unattended as possible. In the eReuse.org architecture, locations are represented by `Place`, which can contain geographical areas or human-encoded keys (“wardrobe number 3”). Places can be as precise as the organization sees fit. On one hand, the preparer can use the eReuse.org

app and the location capabilities (GPS, Wi-Fi) of its smartphone to locate the device in a warehouse. This process is fast and easy, as the preparer just needs to read the QR code from the new tag. In Figure 25 we can see the representation of what can be a warehouse, as seen in the eReuse.org app. On the other hand, the preparer can specify in its DeviceHub the new location, selecting from a list of mapped places. These places can be from warehouses to shelves.

In the system, a user can move a device into a new place by setting the `devices` field of the place, but no record of this is kept. In this step, it is usual too to issue `Receive`, `Ready` or `Locate`, as they add a new record in the life cycle of devices and a meaning to the placement (remember that events can be tied to a place). `Receive` is usually performed by an external employee if devices are set to a warehouse that is not directly controlled by the preparer (another department for example); in this way, the reception is acknowledged by the system. `Locate` is usually done by the same preparer to set a final position for a device in the warehouse. `Ready` has the extra meaning of stating that the placement is final for the device, until it is donated or sold. The benefit of not generating an event is not cluttering the log of devices with many movements, which is interesting if there are only temporary places without real importance in the process.

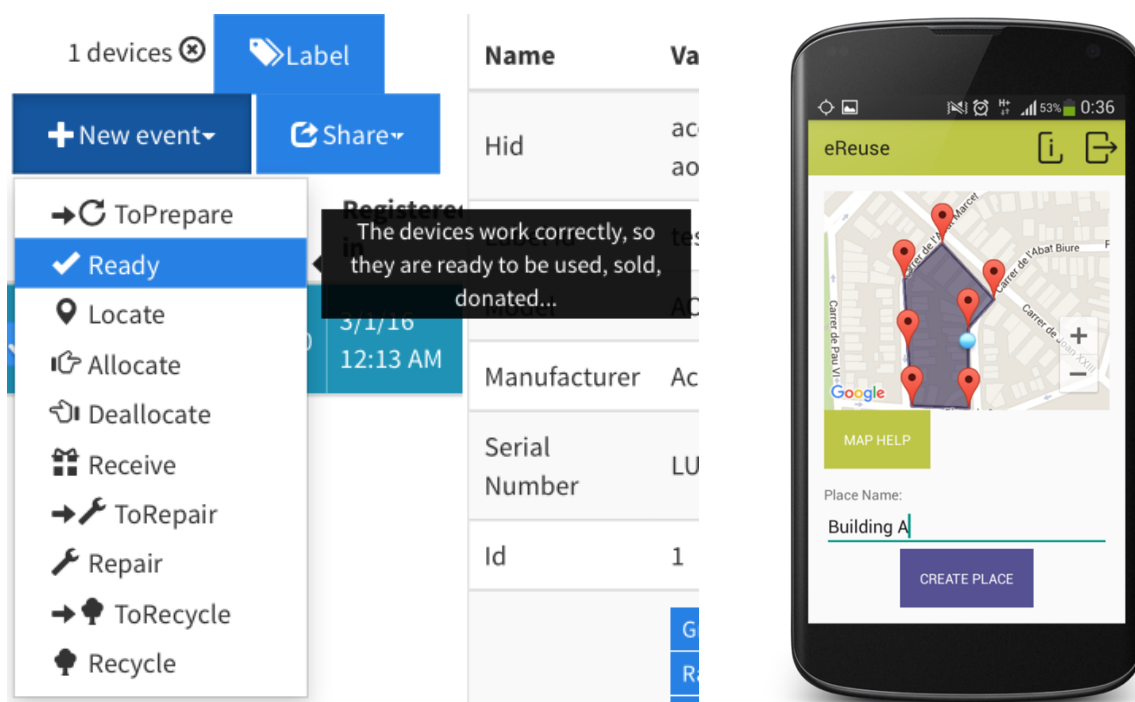


Figure 24 (Left) Screenshot of DeviceHub. We can see the many default events that we can perform on a device. `Ready` (which is the hovered one), `ToRepair`, `Repair`, `ToDispose`, and `toPrepare` are exclusive for the preparation for reuse. Their usage depends on the granularity an organization looks for in controlling this process.

Figure 25 (Right). Interface of the eReuse.org app, where we can see how the user is creating “Building A”, which contains the painted area in grey, with the grey markers as vertexes, and the blue one as the position of the user. This “Building A” can be a warehouse or the location of one client, for example.

5.3.2 The transferal process

In this section, we are going to see the different mechanisms that interact to achieve the transferal, which is where changes the **possession** and the **placement** of a device, from a giver to a receiver.

We can divide transfers using three criteria: the rules involved in the mediation process, the profit the giver seeks, and the existence of intermediaries that keep the device temporarily, by storing them in a warehouse for example (see Figure 26). For the first criterion, transfers can be peer-to-peer (P2P), if donations are direct and there is no control in the transference itself, or they can be on a platform if there are rules imposed by the selling channel. For the second criterion, the transfer can be a donation, if there is no seek for profit, or selling otherwise. Finally, a transfer can be direct if it is delivered from a giver to a receiver, or indirect if the giver gives it to an intermediary, which gives it to the final receiver.

Transferal	Rules involved in the mediation process	P2P
		Platform
Profit the giver seeks for		Donation
		Selling
Existence of intermediary that keeps the device		Direct
		Indirect

Figure 26 Types of transferal by different criteria.

There are different workflows in the transferal process, depending of the criteria explained above and the specific factors varying for each organization. We will explain an exemplifying workflow for devices, which has been co-designed with our actual stakeholders and takes the common elements of their specific workflows and the workflow for projects of the Social Platform Reutilizta.cat. Both workflows are simplified for the sake of the explanation; in concrete, we only keep the essential edges that are travelled when there are no deviations in the process. Figure 29 represents a more complete workflow.

5.3.2.1 Device workflow

The device workflow (see Figure 28) starts with a device that has gone through the Preparation for Reuse process, which means that the device is ready (`Free`) to be reused, and finishes when the device is in possession of the final user. DeviceHub manages the devices in the transferal process, and each state represents an *event* that can be performed in a DeviceHub.

These are the steps:

1. The device is available when there is willingness for it to be sold or donated, by emitting `Free`.
2. The device is assigned to the final user. For a DeviceHub providing devices to a social platform or similar, the manager allocates (`toAllocate`) the device to a final user, and the user will be able to `Accept` or `Reject` the assignment. Note that managers can enforce allocation (without waiting for acknowledgement from the user) by issuing

Allocate directly. After allocation, the device will be **Deallocate** from the giver. For a DeviceHub providing to a selling service, the selling service will automatically emit **Allocate** and **Sell** when a user buys a device.

3. The device needs to travel to the location of the final user. The traceability service of DeviceHub works by locating, with the eReuse.org app or similar, the device when it reaches the end of every stage in the trip. This is, when the first transporter **Receives** the device, it will set so on the system, and this process will be repeated through the different stages until the final user performs the last **Receive** event. For example, if a device passes through three warehouses before it reaches the final user, we will expect at least five **Receive**: one for the transporter that takes the device from the preparer's location, three more when the device gets to each warehouse, and a last event coming from the user's location when the device reaches the destination. If the final user goes to the preparer's location, we will expect just one **Receive**.

The final user receiving the device does not mean that the reuse process has been fulfilled: the device needs to be *used* (5.4.2).



Figure 27. The process of preparation for reuse and transfer by directly donating a device.

1. Registering the device. The resulting metadata of the device.
2. Labelling the device and using the companion app.
3. Finding social support for social projects.
4. Exchanging the device. Users can use the app to acknowledge the donation.

5.3.2.2 Project workflow in Reutilitza.cat

The project workflow of Reutilitza.cat (see Figure 29) is an exemplifying workflow of a TransferHub Program, which works through donations and has, depending of the occasion, indirect and direct channels. As in a DeviceHub, each state represents an *event*²⁶ performed to a project, and many of the events are sent to a DeviceHub or similar. The process is as follows:

²⁶ We do not explain the events of a project as, at the moment of this writing, we are drafting them.

1. A receiver creates a project, which will be a *draft* until a manager accepts it.
2. Once accepted, the project will remain active until it has fulfilled its demand for devices. Note that devices are given in batches, and a batch is usually insufficient to cover the requirements of the project. Meanwhile the projects are active, they are showcased in the website. People can show their support by voting, sharing them to social networks, etcetera.
3. The manager will try to allocate (`toAllocate`) some devices to the project from an ITAMS such as DeviceHub, which will send the event to the actual TransferHub, changing the project from *waiting for assignment* to *devices assigned*.
4. The final user can `Accept` or `Reject` the devices. For TransferHubs with the Selling Services functionality, the user would be required to pay, performing `Sell`.
5. Once the user `Receives` all accepted devices, the project will be closed if it has fulfilled all the devices it asked for, or it will be waiting for a new assignment.

A project can be cancelled at any moment if a manager needs to do so.

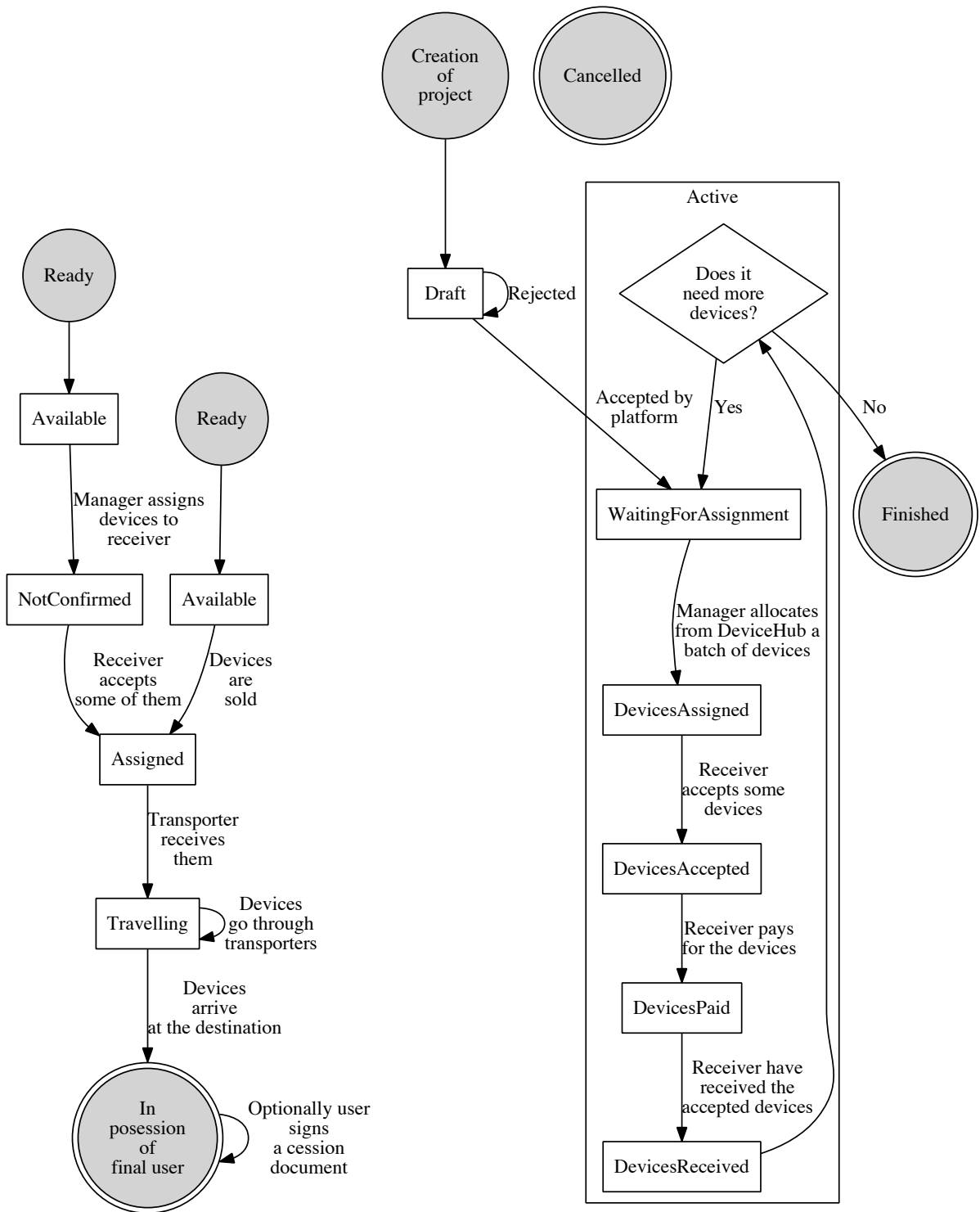


Figure 28 (Left) Workflow for the devices in the transferal process. These actions are performed in a DeviceHub. The state 'Ready' references the final state seen in the last section.

Figure 29 (Right) Workflow for the projects in the transferal process. These actions are performed in a TransferHub. Both systems need to cooperate by notifying each other when an event is triggered in a device or a project. For example, WaitingForAssignment changes to DevicesAssigned when a manager performs the event Allocate from some devices to the specific project. This implies that TransferHub needs to share the projects to DeviceHub.

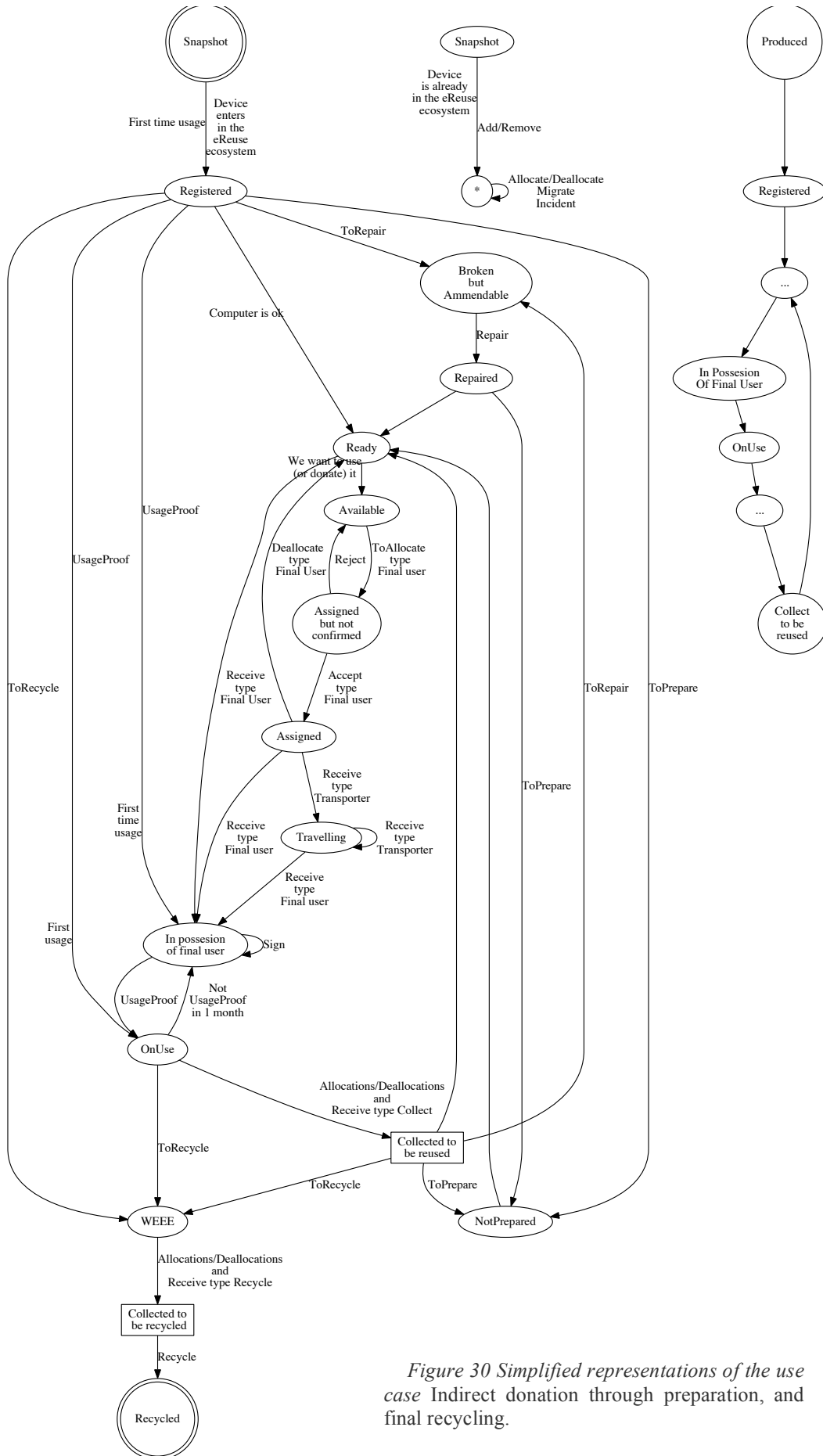


Figure 30 Simplified representations of the use case Indirect donation through preparation, and final recycling.

5.3.3 The collection

This process represents the disposal of a device to an authorized collection point, which can prepare the device for reuse or recycling. The process starts when the device is assigned to be disposed, and finishes when it is received and acknowledged by a certified collection point (see Figure 30).

1. The manager sets the devices `toDispose`, starting the process.
2. Devices are taken to a collection point in a similar way done in the transferal process, with the collection point being the last `Receiver`. To ensure that devices are received to an official recycling collection point, and that they are not illegally sold or thrown away, users can use the geolocation capabilities of the eReuse.org App to perform `Receive`. If the geo-coordinates of the collection point are in the DeviceHub of the user or in the global DCP, the system will be able to compute if the user is inside the area of the recycling collection point, thus adding a layer of verification and efficiency to the process.
3. A malicious technical user would be able to send false geo-coordinates to the system. To fight against this, eReuse.org is collaborating with collection points in Catalonia (Spain) to adopt the technology, in concrete the eReuse.org App; an employee can acknowledge the reception with the app using a smartphone.

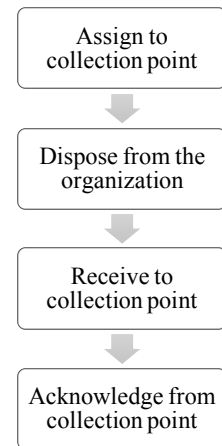


Figure 31
Collection process.

See Section 6.2.5.1 for the implementation in DeviceHub of the collection.

5.4 ENSURING REUSING AND RECYCLING

As we have seen in the introduction, a very high number of devices are lost or pillaged during reusing or recycling. A main issue that eases losses is the lack of control: once a device is found polluting in a place, there is no way to know where the device comes from, and in which step it went out of the process. If we can get this information, we will be able to detect weak spots in the processes, and unmask malicious users—or users that simply do not comply with the eReuse.org standards and even law. This specific control of devices is achieved by tracing them worldwide. To achieve it, we have found the following requirements:

- There must be a way to identify a device (in the following section we will elaborate on this point).
- There is a need for a worldwide way to share, at least, this identifier. Therefore, when a device is found polluting, the same identifier can be generated and, by uploading it to a trusted system, there is a way to obtain the life cycle of the device.
- For this system to work, third parties interested in tracking their devices need to inform the trusted system of, for each device, this special identifier (at least). As more information is added, this system will be better able to analyse weak spots and improve the processes. Note that private information should be leveraged carefully.
- The last statement implies that third parties need to understand and use a shared, standard ontology.

eReuse.org's effort is in convincing governments and organizations to join this venture, as there is a need for them to feed the system with the data of the devices they want to control. This project specifies the Global Record of Devices (GRD) with this purpose: to be a decentralized system that logs traceability metadata about devices, fed by different DeviceHubs and third-party ITAMS. Given the identifier of a device, it can recover its life cycle. GRD asks every device registered for the following two fields: its main URI and the special identifier. From this point, the ITAMS can send as much information as it wants from the device, and it will be used for tracking devices and for generating circularity inputs.

There are some problems in getting an identifier with the expressed characteristics, and we are going to discuss our proposal in the following section.

5.4.1 The Hardware Identifier (HID)

As for traceability needs, a suitable identifier needs to be:

- Unique. There cannot be another device with the same identifier, worldwide.
- Not easily removable. This identifier is going to be most needed when devices are found after abandoning the legal channel. If it has been done by malicious users, we can expect them to complicate the identification of the devices, by for example, removing the tags stuck in the preparation for reuse.
- Easily generated, at least visually, and by using an automated tool. This is so it can be processed in great batches while being visually annotated if the device doesn't work or no software can identify it. The identifier cannot be too long or too complicated, if we expect people to visually read it.
- It cannot relay another software or the Internet, as this process can happen offline.

We can find in the literature specifications about identifying assets. The National Institute of Standards and Technology (NIST) from the United States, defines four ways of identifying an asset: literal identifiers, synthetic identifiers, relationship identifiers and extension identifiers. Literal identifiers are “are the pre-defined fields containing literal values that may identify an asset”, which are the fields generated (or pre-defined) by a manufacturer when the device has been created. Synthetic identifiers “are meant to be used when a database or process assigns an identifier”, which are the identifiers’ organizations and systems assigned to devices when they are introduced to them. Relationship identifiers “are meant to be used when an asset may be identified based on a relationship to another asset”, for example because it is a component of another device or is assigned to one person. Finally, extension identifiers are optional identifiers provided by the manufacturer to help in the identifying process of an asset [49]. Using our already defined list of needs for an identifier, we can discard synthetic and relationship identifiers, as they need to be generated by another system, so it is not inherent from the device itself. Extension identifiers are optional, so we cannot rely on them. Our identifier needs to be literal: we can use the serial number or the MAC²⁷ address to generate our identifier.

²⁷ The Media Access Control address (MAC address) is a unique identifier assigned for network interfaces (such as a network adapter) to be able to communicate with others in a network.

To try this identification method, we got the serial numbers of 1212 computers using an automatic tool (a prototype of DeviceInventory) and the MAC address of the network adapters to test the reliability. We can see the aggregated results in Table (see the file *Serial numbers test.xlsx* in the additional material).

Table 5 Aggregated results of the detection and uniqueness of serial numbers of computers, and some of their components. In the case of the RAM module, we only used one randomly from those available in a computer. Nowadays, the percentage of detection and uniqueness has significantly risen as a result of enhancements; however, we can extract the same patterns.

	Network Adapter	Hard-disk	MotherBoard	RAM	CPU	Computer
Total	1212	1212	1212	1212	1212	1212
Percentage of detected values	99,3	99,3	64,6	97,9	86,6	87,0
Percentage of unique values	100	97	93	15	3	97

We can see how the detection for an overall computer is high; however, 13% of them were not detected. This is the usual case when the computer is assembled by an individual, by buying and mounting the components, although it still can happen with few manufactured computers²⁸. For the percentage of RAM, although the detection is high, the value is not unique, as different manufacturers use the same serial number to identify different modules. Finally, for the CPU, it is not as unique as Intel and other companies that create processors removed the serial number (or they never added it in the first place), and the data that returns is meaningless, it is believed due to strong criticism against privacy²⁹.

As a conclusion, we cannot identify CPUs; however, we are able to do so with all the other components, although not always. We need to deal with devices that cannot generate value (the non-detected ones); however, the real problem is dealing with non-unique or invalid values, as although we can detect collisions when registering them in a system connected to the Internet (GRD, for example, would detect them), it cannot be done when dealing with them offline.

To solve these problems, we take a number of identifiers and other metadata that is usually present in a device. By concatenating them, we create an identifier that has proven not to collide. The result is the Hardware Identifier (HID), which is the global identifier used by eReuse.org to uniquely identify a device. The HID never changes for a device, and it can always be obtained automatically by scanning—visually or using a software, if possible—from the device.

The HID is the concatenation of, in the exact order: the manufacturer name, the serial number and the model, joined with hyphens, and adapted to comply with the URI specification, so it

²⁸ Inexplicably, we cannot extract serial numbers from few computers that are the same (manufacturer and model) than others that behave correctly. It is future work in researching and solving this fact.

²⁹ For example: <http://cryptome.org/stoa-r3-5.htm>

can be used in the URI identifying the device on the Internet. The conversion is done as follows: (1) non-ASCII characters are converted to their ASCII equivalent or removed; and (2) characters that are not numbers or letters are converted to underscores, in a way that there are no trailing underscores and no underscores together. The result is represented by the following *regex*:

Listing 3 Simple regex representing the HID (it could be more precise).

```
[\\w]+-[\\w]+-[\\w]+
```

Listing 4 Example of HID, representing the computer Acer with model aod270 and serial number lusga_0d0242201212c7614-aod270

```
acer-lusga_0d0242201212c7614-aod270
```

The HID forces the uniqueness constraint of an identifier, at a cost of potentially generating lesser identifiers, in the cases where the model or the manufacturer names are missing.

Finally, we still have many devices that cannot generate a HID. We cannot guarantee the traceability of these devices. However, we can use several mechanisms to try to identify—with a margin of error—a device, using the other identifiers proposed by NIST. We can use synthetic identifiers when organizations use an internal identifier for each of their devices, although they are not usually used for components. In this case, we can use relation identifiers, identifying components with the devices that contain them. Finally, extension identifiers, alongside other immutable metadata (the number of cores of a processor, or the maximum speed of a network adapter for example) can be used when identifying a device to compare if there is a similar one already registered, and try to guess if they are the same.

DeviceHub and GRD implement some of the mentioned mechanisms to try to identify devices that cannot generate a HID. This is why it is important to retrieve as much metadata as possible, as there are more arguments when determining if two devices are the same. DeviceHub has more capacity to perform this than GRD, as DeviceHub will always have more information than GRD, and ultimately can ask to a user for validation. In Section 6.2.3.3, we describe the implementation in DeviceHub.

5.4.2 Acknowledging final usage

The transferal process finishes when the final user receives the device. However, the reused cycle is not completed until the user *uses* the device. To acknowledge this, we provide a small and non-intrusive tool that can be installed by the preparer in the computer. This tool connects periodically (monthly) to the Internet, firing a kind of *Keep Alive* signal to a specified ITAMS. This signal just sends the information to identify the device, and it is a way to know if the user has, at least, turned on the device. If a computer has not sent this signal for a large period of time (two years for example), the ITAMS could be used to contact the final user asking about the computer, as maybe the user has not recycled it correctly or is not using it because it is obsolete for the user.

5.4.3 Tracing devices

Traceability is key to ensure reusing and recycling. Devices need to move through locations, and they can be leaked or stolen. eReuse.org can battle against this by providing accurate

information, such as where the device is in different stages and in different times, so when an irregular action happens we can get the last known position and user. Traceability needs the following to work:

- An URI with a HID. Devices without HID will not be fully traceable, as seen in Section 5.4.1.
- The position of the device. This can be done by scanning the stuck tag of the device with the eReuse.org App, or by using location IP Servers. A computer with installed small software can periodically send its IP, and an IP location provider can locate, at city level precision, the position of that computer.
- Reporting to a global trusted system such as GRD and DCP, so users can share strategic positions in the reuse and recycle processes, such as collection points and traceability information about their devices. Sharing traceability information is key when a user recovers a lost device (contaminating for example) and wants to find its last owner.

For devices, Device Diagnostic and Inventory collects metadata about devices and generates a report that is uploaded to a DeviceHub; the DeviceHub sends traceability information to GRD. GRD expects the following traceability information for any device: (1) the `@type` of device, (2) the URI, (3) the HID if exists, (4) synthetic identifiers if any, and (5) optional immutable metadata.

For events, it is expected but not required to send them soon after they have been performed. GRD expects the following events to be sent: `Register`, `Deallocate`, `Allocate`, `UsageProof`, `Receive`, `Add`, `Remove`, `Locate`, `Recycle`, `Migrate`. The minimum required information for the events are: (1) the URI, (3) the URI of the user who performed the event, (4) the date of the ITAMS when the event was triggered, (5) the URI of the affected device, (6) the `@type` of event.

For places, users create places into DeviceHub (mapping their coordinates), and if the user sets the place as public, DeviceHub sends basic information to DCP. DCP requires the following information: (1) the URI, (2) a human name or `label`, (3) the `@type` of place, (4) a polygonal area represented in GEOJSON (`geo`). In Figure 12 we can see a mapped collection of points in Barcelona retrieved from DCP.

5.4.4 Use case: user finds a leaked device

If a user finds electronic waste that has been monitored by eReuse.org, it can use GRD to notify about the problem by sending its location and writing some information, so GRD can warn a manager from the last ITAMS *holder*. If the last known position and owner of a device is a recycling collection point, we can assume this device was leaked in the collection point. If the last user was a final user, maybe this user did not properly dispose of the device, or it did not inform about it.

Leaked devices may not be in good conditions, so there are many ways to identify the device in GRD: (1) scanning the QR code in the printed label with the eReuse.org app, (2) introducing the HID (if printed in the label) at the GRD website, (3) executing Device Diagnostic and

Inventory, and (4) reading the serial number, manufacturer, and model names from the incorporated tags by the manufacturer.

By firing an event not coming from the *holder* of the device, GRD will detect an exception and will warn the last ITAMS *holder*, creating a channel for both the *holder* and the user who found the device. The final user can access to the public information of the device existing in GRD, and it will be up to the *holder* ITAMS to deal with the issue by providing more info, contacting the final user, etcetera.

5.5 EXTENDING AND INTEGRATING eREUSE.ORG

We have seen how many parts of eReuse.org are more effective when there are more users and third parties involved, such as computing circularity inputs or ensuring traceability. We justify in Section 5.2 how the vocabulary and API fit this purpose. In Section 6.2.10, we see how DeviceHub is easily extendable and can be integrated with other systems.

5.6 MAXIMIZE THE SOCIAL USAGE

In this section, we define the social usage of a device and how eReuse.org helps in maximizing it.

If we state that the social usage of a device is the value generated to society through its lifetime³⁰, we could mathematically define it as follows:

$$u = v_1 + v_2 + v_3 + \dots + v_n = \sum_{i=1}^n v_i$$

Where u is the social usage, v_i the value generated in a moment of time, and n the lifetime of the device. To increase the social usage, we should increase n or v . Increasing lifetime means making the device last longer. There are many ways to achieve this:

- By making the device more durable, which means that the components are built with quality, and programming obsolescence is removed from the device.
- By refurbishing devices, fixing, and replacing components. This is done in the preparation for reuse, and DeviceHub, TransferHub, and GRD record the number of times it is done per device.

Increasing the value of a device in a given time means using it in a way that maximizes the return to society. Ensuring this is a difficult task, as it depends on the usage of devices: from the task the user is doing, to the efficiency of the user in performing it. These things are difficult to measure and compare, as they are influenced by culture, study level, etc. The eReuse.org system assists communities in selecting appropriate receivers, and helps auditors and

³⁰ Note that this term is similar to the circularity indicator of Utility during the use phase; however, we do not measure the intensity of the usage, but the value society can extract from it.

communities monitor usage, which is one of the principles for the eReuse.org commons [12]. These are the ways:

- Finding good candidates for a device. Creating, reusing, and recycling a device is an investment of resources. To maximize the return of value, eReuse.org lowers reusing and recycling costs, for example, making the different processes involved more efficient. One way of reducing the cost of the transferal process is by finding suitable receivers that are nearby, so transportation costs can be kept low. Another way is by increasing the value for the candidate:
 - With the benchmarks performed by DDI (CPU capabilities, for example), and with the knowledge of the necessities of the receiver, a recommender system can find a match between offer and demand in terms of computational power. This is actually done manually by a manager in Reutilitza.cat, as receivers state the usage of the devices they need. For example: Reutilitza.cat keeps the most powerful computers for projects that want to do graphical design, and assigns less powerful ones to users that want to use office tasks.
 - Monitoring the reputation of the receivers. eReuse.org can annotate if receivers do not tend to finish the projects they create, or they do not recycle and reuse properly. It is future work to create a reputation system, integrating it with the existing ones. In this way, the system has more parameters to know the value of a receiver.
 - Evaluating the social impact of a receiver. How much the society can benefit from a receiver? It may be hard to answer; however, it is easier to report few indicators explaining the impact of use, such as the number of people trained, activities performed, etcetera. For example, Reutilitza.cat demands the manager of every project to write a report months after the reception, stating how the devices have helped and the impact they had on the project, and ultimately the impact the project had on society. This helps Reutilitza.cat in making future decisions for similar projects or for projects where the same people are involved; it also shows donors the value of their donations.
- Increasing usage-time. Lifetime is the time a device is operational; usage-time is the time the device is used. A device can have a long lifetime because it has never been used, so it is not deteriorated; however, the value this device brings to society is none. Measuring usage-time is difficult: for example, users can leave computers turned on without actively using them, or they might be performing meaningless tasks. There is software in the market able to monitor them precisely, at the expense of losing privacy; it is not an option for this project. We propose an initial step: extending the software capable of acknowledging final usage (Section 5.4.2) to provide the aggregated amount of time the device has been on, balancing precision and privacy. This data can be assigned to the user that **Received** the device until disposal.

5.7 PROVIDING INPUTS FOR CIRCULARITY

Generating data for measuring circular indicators enables a transition towards a fully circular economy, as it builds the knowledge base for environmental action and sustainability. GRD and DCP receive public information from other systems such as DeviceHub or TransferHub, which can be used to ensure traceability, or to provide input for circularity indicators. We present an initial proposal of inputs GRD and DCP can provide:

1. The number of times a device has been reused, the sum of all acknowledged usages after a successful transferal.
2. Durability: elapsed time between first usage and the last one. This is computed after collecting the device for recycling.
3. Running time: the total time the device has been operating. This information is provided automatically by users to platforms using programs installed on devices. To preserve the privacy of users, eReuse.org only collects the total time.
4. The usage-time, as described in Section 5.6, and the Material Circularity Indicator (MCI), which we can obtain when comparing usage-times of devices [10].
5. The performance of a device, as described in Section 5.6.
6. The destination after use: “How much material goes into landfill (or energy recovery), how much is collected for recycling, which components are collected for reuse” [10].

6 DEVICEHUB

This project generates a software solution to manage the reuse process efficiently, ensuring disposal. In this section, we explain DeviceHub (introduced in Section 5.1.2): the requirements, the implementation and the API. Note that the vocabulary of DeviceHub (with the class diagrams) is explained in Section 5.5.2.

6.1 REQUIREMENTS

1. To register, access, manage, process, and delete the metadata of devices. It includes information to identify a device and its components uniquely, to be able to generate circularity data and to ensure reusing and recycling.
2. To manage users by providing a permission system. Permission systems decide the operations that users can perform based on arbitrary parameters. As an example, the owner of a device and the administrator should be the only ones with the ability to update its components.
3. To perform events. Events are intrinsic parts of the life cycle of a device.
4. To manage places of special interest in the process of reusing and disposing. Events happen in places. By knowing them, we can help in guaranteeing circularity and in optimizing reusing.
5. To produce a tagging system, so devices can be labelled. QR codes need to be generated after registering the devices' process so they can be easily managed.
6. To interact through the eReuse.org ecosystem in a decentralized architecture. To be able to effectively reuse and recycle, organizations need to share data, and the other systems of eReuse.org need to cooperate.
7. To issue certificates regarding reuse and recycle, such as erasure certifications, etcetera. These certificates will need to ensure veracity of data. Thus, DeviceHub will need to implement encryption, signing, and validation mechanisms.
8. To be easily extendable and integrable by third parties. This enforces a modular system adopting extensions (plug-in), and dividing the interface (DeviceHubClient) with the data server (DeviceHub).

6.1.1 Target users

DeviceHub is instantiated at an organization level (which could be a full company or a department). For every organization, we can identify the following target users:

- Extraneous users: Users that are not part of the organization, but need to view some devices in the system and in some cases update their state. For example, auditors or final users, such as clients or regular employees. Final users can perform some

actions, such as communicating when devices do not work, so technicians can repair them.

- Technicians working in the ITAM: they are the ones registering devices and working with them. They can be preparers, working in a collection point, project managers, etc.
- Administrators: Users in charge of the operational part of the ITAMS who need to control the system, such as high-level managers.
- Super users: Computer engineers who maintain and/or develop the ITAMS, and need to have full control over the system.

6.2 IMPLEMENTATION

In this section, we explain the technologies used in DeviceHub, the development model, the philosophy behind the design, and how it achieves every requirement.

6.2.1 Technologies used, development model and design

DeviceHub is a dual project: a server framework extending Eve and a client app extending Angular (see Figure 32). Users, using their browsers, first connect to the server where the DeviceHubClient is, obtaining the code to load the web client. Once loaded, DeviceHubClient performs REST operations against the API of a pre-assigned DeviceHub.

DeviceHub offers an isolation mechanism per organization, so different organizations can use the same instance of DeviceHub but still have a database of its own. With the help of the DeviceHubClient interface, the user has the illusion that its organization is the only one in the system. This helps enhance reliability, privacy, security, and performance.

- The organization can have the database in an isolated machine, reducing the chance of being infected by other databases in the system, if compromised.
- The organization can have total control of the database, and this can be inside their servers or in a country of choice.
- The servers supporting databases can add more resources per database, as they have lesser ones to support.

A DeviceHub instance keeps a special database where it stores the accounts of all its users, regardless of where the data of the organization is. This is made:

- To ease the interaction with the user: if there is no centralized database to look up an account, the user would need to introduce the database to look for when logging in, adding an extra step of difficulty, which is one of the main problems of similar systems such as OpenID [50].
- To enhance integration through all the eReuse.org ecosystem, it is desired to have some common OpenID-like system (without the flaw commented above) to authenticate a user through the different systems. It is future work to develop such a system, and it is a first step to provide it in the same instance of a DeviceHub.

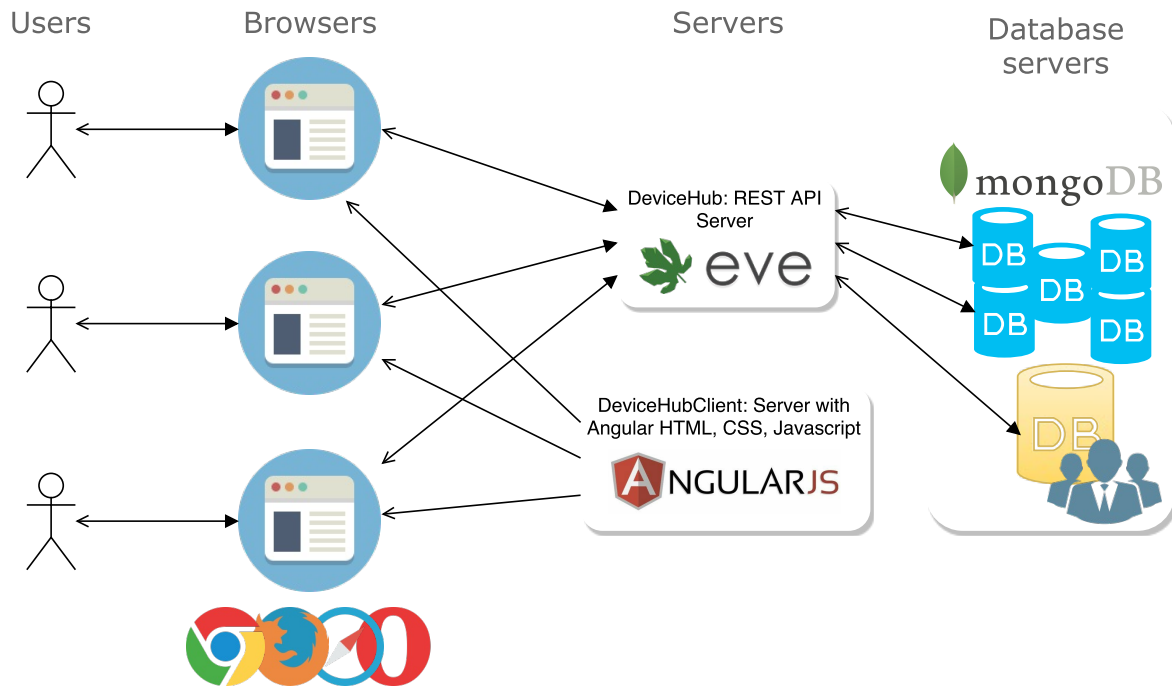


Figure 32 Diagram representing the technologies of DeviceHub and DeviceHub client, and how they interact with the user and with each other. DeviceHubClient supports the latest versions of Chrome, Firefox, Safari, and Opera.

6.2.1.1 Design of DeviceHubClient

After trying different ITAMS, we arrived at the conclusion that they are generally complex: they offer too many options and configurations, which may be handy for specialized and trained users, but it is a barrier for unspecialized and newcomers. As our objective is bootstrap reusing, DeviceHubClient has to be built with simplicity in mind: easy to manage basic reusing, and more personalized for technicians. Finally, being simple is a statement and a differentiation from the rest of the tools of the market.

DeviceHubClient feels new, responsive, and fast. Circular economy and reusing are new concepts we introduce in ITAMS, and the design helps in transmitting this idea. We use the open-source theme for Bootstrap, Cosmo, which is “an ode to metro” [51], referencing the new design style introduced by Microsoft in 2010 (this is now called *modern design*) [52]. At the same time, we use light animations such as transitions when pressing some buttons or showing modals³¹, adding dynamism to the user interface; we use the HTTP cache to speed-up information retrieval.

The first screen the user sees when entering a DeviceHubClient is the login screen, which is only a logo and a form with two fields, a checkbox, and a button—the minimum required information to login.

³¹ “Modals are HTML elements that are hidden in a web page, sliding down from the top of the screen when triggered”. For example, it is used to display dialog prompts, such as alert and confirm dialogs [74] (see Figure 37).

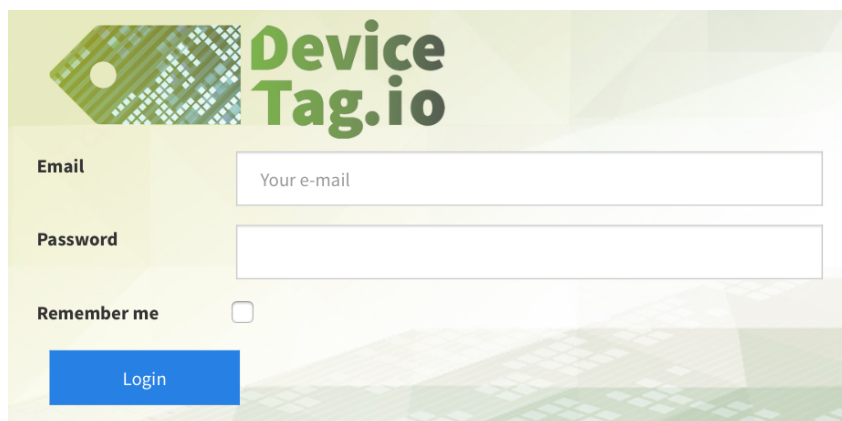


Figure 33 Login screen for DeviceTag.io.

Upon logging in, the user sees the main screen from DeviceHubClient. Part of the simplicity of DeviceHubClient is the fact that there are only these two screens. You can navigate through the two tabs on the left, changing the content inside (which we call a sub-view). The menu on the top right represents the main menu, consisting of regularly used options: registering new devices, download links for DDI, and the eReuse.org App, account management (with an option for logging out), and a database selector.

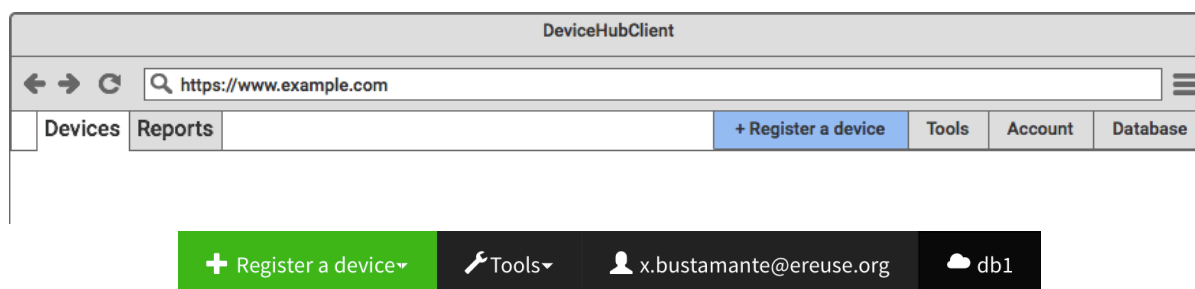


Figure 34 (Top) Mock-up representing the main view in. (Bottom) Representation of the right top menu in DeviceTag.io.

By clicking on the *Devices* tab, you go to the management sub-view (see Figure 35). It is shown by default; in this sub-view, the user can manage the devices. It is divided by a three-column design (from left to right): a list of places, being able to manage or create them; a list of devices, being able to search, select and work with them; and a view showing the selected device of the list. The last view can be alternated, by showing the metadata of the device, or a log of events. The user can filter devices from the list by using the search or clicking in a place from the place list.

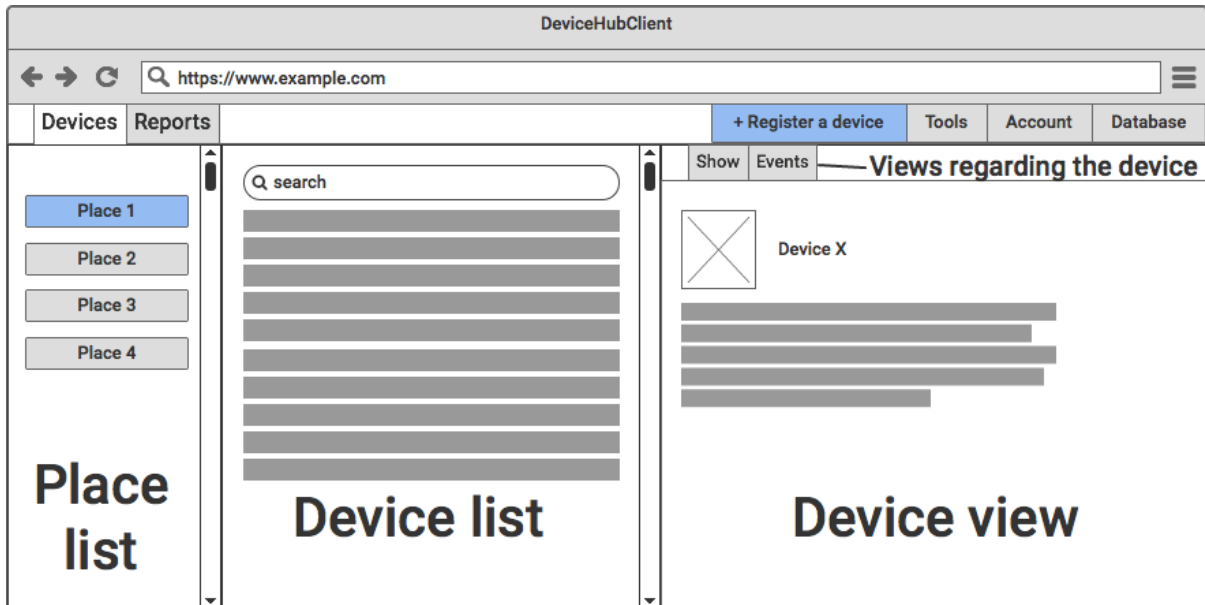


Figure 35 The three-column design of the sub-view for managing devices.

In the device list, at the left of every device there is a big checkbox (Figure 36). By pressing it, the user enters into the device management: a horizontal menu appears with the options the user can perform on the selected devices. This is a familiar behaviour used by other programs, such as Gmail.

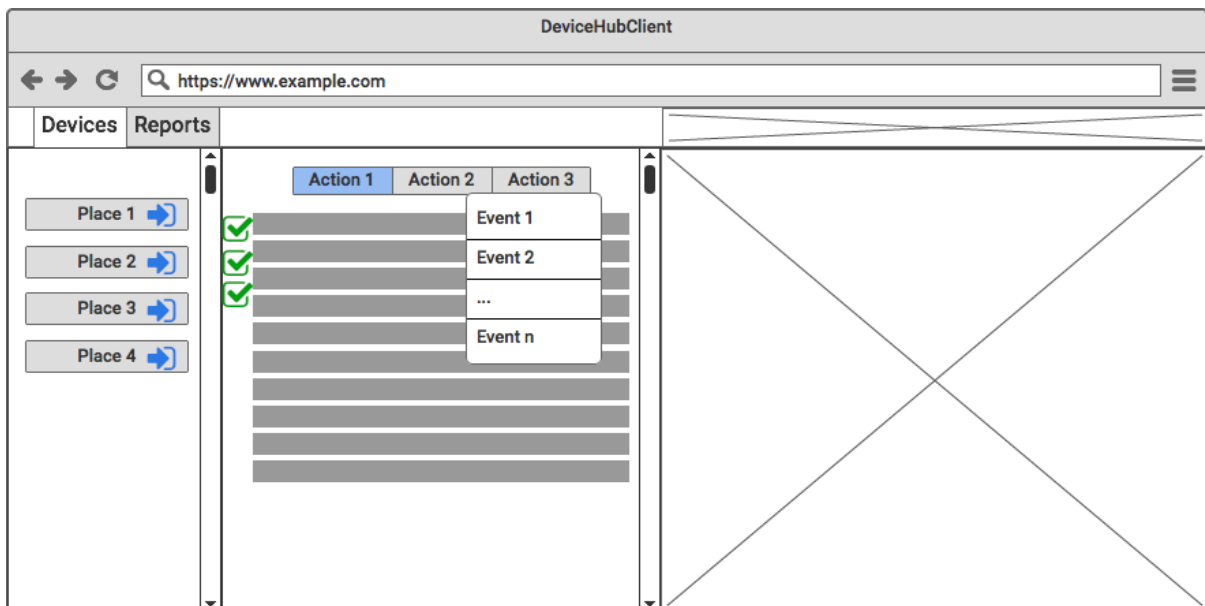


Figure 36 The UI reacts when the user clicks on one of the checkboxes of the device list, showing possible actions.

Some actions and buttons need more interaction from the user, such as submitting a form. In these cases, they generate a modal, covering part of the main view, as if it were a new window (see Figure 37). By completing or cancelling the task, the modal will disappear and the user will be again in the place they were before being shown the modal.

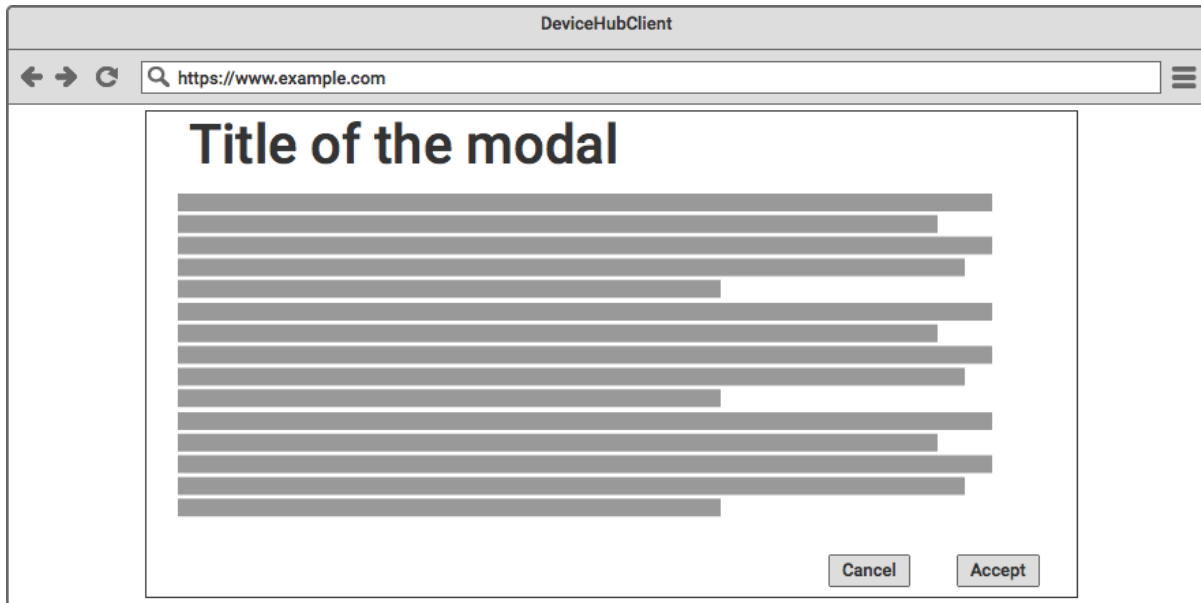


Figure 37 The modal.

We have performed informal usability tests with stakeholders. Thanks to their feedback, we have developed many functionalities, such as a tooltip enumerating the selected devices in the device list (as users can apply new filters and visually lose some devices, see Figure 38), or previews of embedded resources (see Figure 39).

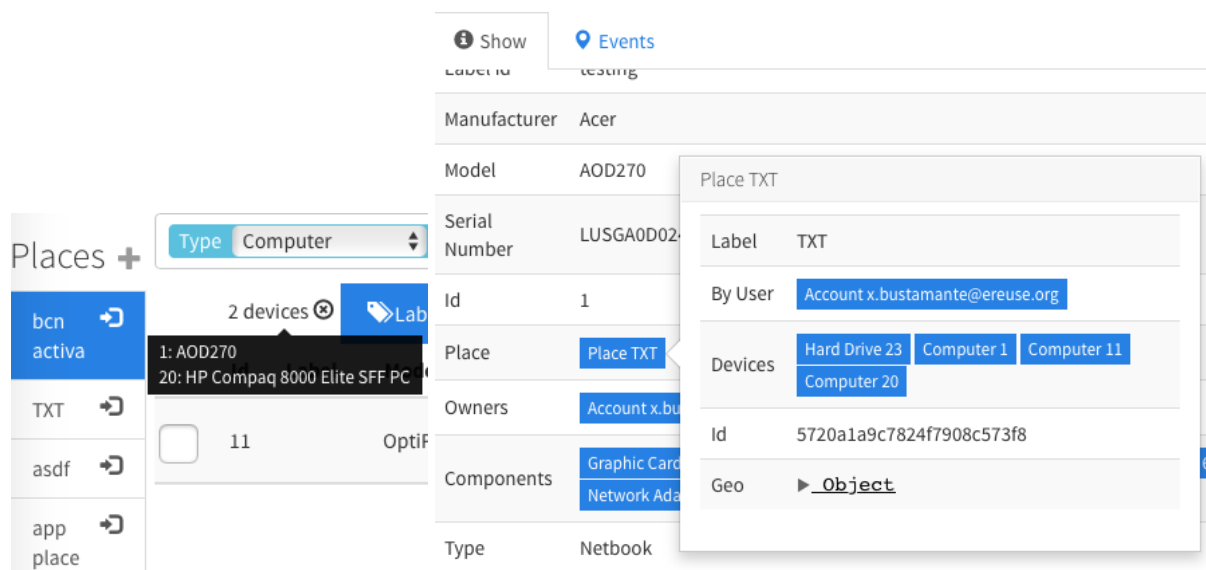


Figure 38 (Left) When selecting devices, users can apply filters that make already-selected ones disappear from view. To ensure they are still selected, there is a counter of devices that, upon hovering the cursor, shows a list of selected devices.

Figure 39 (Right) The preview of a resource (the floating box in the image) is useful when the user wants to check out the resource (in this case information about the Place 'TXT'), without having to lose the task it was doing.

There is still a long path for usability and interface design, as we need to polish the experience, perform usability tests and develop some aspects that we have not been able to do because of timing constraints, specifically with details. Future work is using the HTML5

Notification API to develop a notification mechanism, so users can get notified when assigned tasks (for example, when they need to repair a device); and creating an interactive tutorial training the user with the most important aspects.

6.2.2 Installation and configuration

6.2.2.1 DeviceHub

The requirements to install DeviceHub are:

- The latest versions of Python 3.4 or 3.5.
- MongoDB.
- The installer pip.

Type in a console `pip install eReuse-DeviceHub`. And then to use it, create a Python file and write the following:

Listing 5 Basic instantiation of DeviceHub.

```
from ereuse_devicehub import DeviceHub
app = DeviceHub()
if __name__ == '__main__':
    app.run()
```

You can configure DeviceHub. Look at <http://python-eve.org/config.html> to learn how to do it, and in the Settings section of the Annex you can see the extra parameters DeviceHub uses.

Listing 6 Recommended minimum configuration for DeviceHub. It sets two databases, the own URL, the URL of DeviceHubClient, and an account for the GRD module (see Section 6.2.8), as this module needs an account to access DeviceHub (which is created if it does not exist).

```
DATABASES = 'db1', 'db2' # We configure as many databases as needed. This is the
name in the URL.
DB1_DBNAME = 'dh_db1' # We link the name in the url with the name of the database
DB2_DBNAME = 'dh_db2'
CLIENT = 'http://my-devicehub-client.com' # For redirections. No trailing slash
BASE_PATH_SHOWN_TO_GRD = 'https://my-own-domain/' # Once you send the first
device to GRD, always maintain this. Just affects at GRD.
GRD_ACCOUNT = {
    'username': 'ereuse',
    'password': 'ereuse@grd'
}
```

6.2.2.2 DeviceHubClient

To install DeviceHubClient, download the release from X. Open the file `bundle.js` in the folder `js` with a text editor, and look for the text `www.example.com`. The first result is going to be in an in-lined JavaScript object similar to the following:

Listing 7 Configuration object for DeviceTag.io.

```
{
  appName: 'DeviceTag.io',
  url: 'https://api.devicetag.io',
  deviceInventoryUrl: 'https://github.com/eReuse/device-inventory/releases/latest',
  siteLogo: 'common/assets/devicetag-logo.png',
  showSiteLogo: true,
```

```

androidAppUrl: 'https://play.google.com/store/apps/details?id=org.ereuse.scanner',
html5mode: true, //If true, uncomment the line in the header in index.html
eReuseLogo: 'common/assets/ereuse-logo.png',
loginBackgroundImage: 'common/assets/login-background.jpg',
headers: {
  "Content-Type": "application/json",
  Accept: "application/json"
}
}

```

The minimal configuration is setting `url` with the reference to the desired DeviceHub. It is recommended that `html5mode` is left on, but it may be necessary to turn it off if the server of DeviceHubClient is not configured to use the HTML5 History API. You will need a server such as Apache 2 to serve DeviceHubClient. Use the following configuration as an example:

Listing 8 Apache2 configuration for DeviceHubClient. Note that you may change the port, the server name and the two paths.

```

<VirtualHost *:80>
  ServerName www.example.com

  DocumentRoot /absolute/path/to/project/folder

  <Directory /absolute/path/to/project/folder>
    Order allow,deny
    Allow from all
    Require all granted

    <IfModule rewrite.c>
      # Optional. HTML5 mode
      RewriteEngine on

      # Don't rewrite files or directories
      RewriteCond %{REQUEST_FILENAME} -f [OR]
      RewriteCond %{REQUEST_FILENAME} -d
      RewriteRule ^ - [L]

      # Rewrite everything else to index.html to allow html5 state links
      RewriteRule ^ index.html [L]
    </IfModule>

    <IfModule mod_expires.c>
      # Optional. Cache control
      ExpiresActive On
      ExpiresDefault "access plus 2 hours"
    </IfModule>
  </Directory>
  #LogLevel debug
</VirtualHost>

```

6.2.3 Managing metadata of devices

Devices are the main resource in DeviceHub, and managing them is the most important requirement for succeeding in reusing and recycling.

6.2.3.1 Processing snapshots

Snapshot is the default event when updating devices. Its parameters are the actual state of a device, this is, all the metadata regarding the device and its components, and updates the system to reflect the state. The system update is done via executing the following events: Register, Add, Remove, TestHardDrive, EraseBasic, and EraseSectors. The execution of Device Diagnostic and Inventory (DDI) results in the generation of a Snapshot JSON file, which is uploaded to a DeviceHub. The workflow of a Snapshot is as follows:

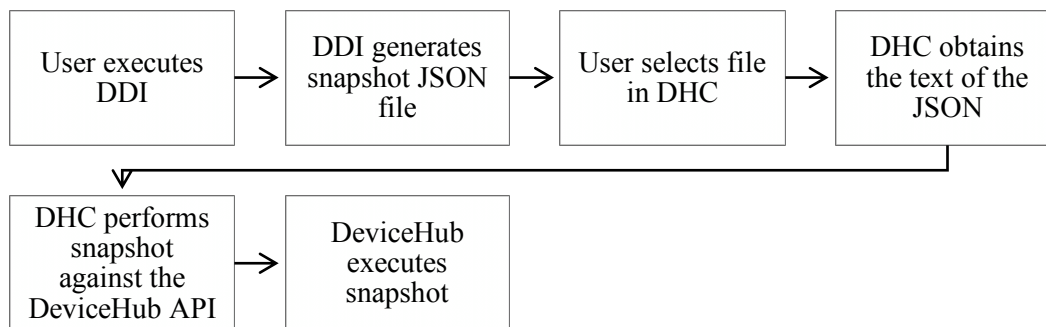


Figure 40 Workflow of a Snapshot, from executing through DDI until it is fired in a DeviceHub. DHC stands for DeviceHubClient.

Snapshot has the following parameters (see them in detail in Section 5.2.2.3.2 and [Section Snapshot API], and Listing 9 for an example):

- A Device.
- An unordered list of Component. They can have Benchmark fields, and HardDrive can have Test and EraseBasic or EraseBySectors. Snapshot will take these events and fire them.
- Information regarding the Snapshot, such as an identifier for the label, comments, or the version.

Listing 9 Illustrative Snapshot.

```

1. {
2.   "@type": "Snapshot",
3.   "components": [{
4.     "@type": "GraphicCard",
5.     "manufacturer": "Intel Corporation",
6.     "memory": 256.0,
7.     "model": "4 Series Chipset Integrated Graphics Controller"
8.   }, {
9.     "@type": "GraphicCard",
10.    "manufacturer": "Intel Corporation",
11.    "model": "4 Series Chipset Integrated Graphics Controller"
12.  }, {
13.    "@type": "HardDrive",
14.    "benchmark": {
15.      "@type": "BenchmarkHardDrive",
16.      "readingSpeed": 143.0,
17.      "writingSpeed": 48.9
18.    },
19.    "interface": "ata",
20.    "model": "SAMSUNG HD254GJ",
21.    "serialNumber": "S27LJ9AZ404021",

```

```

22.     "size": 238475.1796875,
23.     "test": {
24.         "@type": "TestHardDrive",
25.         "error": false,
26.         "firstError": null,
27.         "lifetime": 14320,
28.         "status": "Completed without error",
29.         "type": "Short offline"
30.     }
31. }, {
32.     "@type": "RamModule",
33.     "interface": "DDR3",
34.     "manufacturer": "JEDEC ID:80 CE",
35.     "serialNumber": "C3146863",
36.     "size": 2048,
37.     "speed": 1333.0
38. }, {
39.     "@type": "Motherboard",
40.     "connectors": {
41.         "firewire": 0,
42.         "pcmcia": 0,
43.         "serial": 0,
44.         "usb": 8
45.     },
46.     "manufacturer": "Hewlett-Packard",
47.     "model": "3646h",
48.     "serialNumber": "CZC0230NYM",
49.     "totalSlots": 0,
50.     "usedSlots": 1
51. }, {
52.     "@type": "NetworkAdapter",
53.     "manufacturer": "Intel Corporation",
54.     "model": "82567LM-3 Gigabit Network Connection",
55.     "serialNumber": "00:23:24:05:5d:da",
56.     "speed": 1000
57. }, {
58.     "@type": "OpticalDrive",
59.     "description": "DVD-RAM writer",
60.     "manufacturer": "hp",
61.     "model": "DVD A DH16AAL"
62. }, {
63.     "@type": "Processor",
64.     "benchmark": {
65.         "@type": "BenchmarkProcessor",
66.         "score": 21280.2
67.     },
68.     "manufacturer": "Intel Corp.",
69.     "model": "Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz",
70.     "numberOfCores": 4
71. }, {
72.     "@type": "Processor",
73.     "benchmark": {
74.         "@type": "BenchmarkProcessor",
75.         "score": 21280.2
76.     },
77.     "numberOfCores": 4
78. }, {
79.     "@type": "Processor",
80.     "benchmark": {
81.         "@type": "BenchmarkProcessor",
82.         "score": 21280.2
83.     },
84.     "numberOfCores": 4
85. }, {
86.     "@type": "Processor",
87.     "benchmark": {

```

```

88.     "@type": "BenchmarkProcessor",
89.     "score": 21280.2
90.   },
91.   "numberOfCores": 4
92. }, {
93.   "@type": "SoundCard",
94.   "manufacturer": "Intel Corporation",
95.   "model": "82801JD/DO (ICH10 Family) HD Audio Controller"
96. }},
97. "device": {
98.   "@type": "Computer",
99.   "manufacturer": "Hewlett-Packard",
100.  "model": "HP Compaq 8000 Elite SFF PC",
101.  "serialNumber": "CZC0230NYM",
102.  "type": "Microtower"
103. },
104. "label": "176",
105. "version": "7.0.3b6"
106. }

```

The execution of a Snapshot is as follows:

1. Register the devices that are required.
2. Generate a list of events to execute, taking care of permissions and other criteria.
 - a. Add event executes when an already existing component that it was not inside the device is found in the Snapshot.
 - b. Remove executes when a component that was inside a device is not found in the Snapshot.
 - c. TestHardDrive, EraseBasic, and EraseBySectors are fired if a field test or erasure is found respectively (the @type field in an erasure defines the type of erasure).
3. Execute the events listed in 2.
4. Save the Snapshot event in the database with a reference of the devices and events triggered.

To illustrate the logic behind the execution of Snapshot, we present the following scenario:

(1) The database of the system is empty. The user executes the first Snapshot, which contents are represented on the left column.

Devices appearing in first Snapshot	Conclusions arrived at by DeviceHub	Results
Computer with HID 1	There is no device with HID 1. I need to create it.	Register the devices.
Component GraphicCard with HID 2	There is no device with HID 2. I need to create it.	
Component HardDrive with HID 3	There is no device with HID 3. I need to create it.	
Component Processor without HID.	The processor does not have HID, but I can identify it using the computer parent device. As the computer is new so is the processor.	

(2) Sometime after, there is the following Snapshot:

Devices appearing in second Snapshot	Conclusions arrived at by DeviceHub	Results
Computer with HID 1	This HID already exists.	Nothing.
Component <code>GraphicCard</code> without HID.	The computer already had a <code>GraphicCard</code> , but that one had HID, so they cannot be the same.	<code>Register</code> a new <code>GraphicCard</code> with the <code>Computer HID 1</code> as <code>parent</code> .
Component <code>HardDrive</code> with HID 3, with one test and one basic erasure.	The <code>HardDrive</code> HID 3 exists and it is in the same computer as HID 1. I see one test and one erasure.	Create a new event <code>TestHardDrive</code> and a new event <code>EraseBasic</code> for the component.
Component <code>Processor</code> without HID.	I cannot guarantee it; however, a <code>Processor</code> without HID and with the same characteristics existed within this computer. I suppose they are the same.	Nothing.
(missing <code>GraphicCard</code> with HID 2)	This computer should have a <code>GraphicCard</code> with HID 2, which is not listed in this Snapshot. This means this component is missing.	Remove the <code>GraphicCard</code> from the Computer.

6.2.3.2 The Register event

The `Register` event creates a new device. If the device has components, it will create only the components that did not exist before. This method returns the references of the device and all its components, new or not. The event is usually triggered in a `Snapshot` event, or manually through `DeviceHubClient` or any other client. A simplified workflow is as follows:

1. `Register` performs `POST /devices` with the device in the `device` field. If this fails, it is aborted.
2. `Register` performs `POST /devices` with every component. If the creation fails because the component already existed, the execution continues. The process is aborted if there is another type of failure.

6.2.3.3 Creating devices

Devices are directly created using the default REST interface: `POST /devices`. However, this is supposed to be an internal method and it is blocked for regular users. The recommended way is creating a `Register` event, as it considers more things, and it can be sent to GRD and other systems.

The most complicated part of creating a device is identifying whether it is a new or existing one. There are many sophisticated ways to do this, and we plan to implement them in the future. At this moment, the actual implementation is:

1. DeviceHub tries to create a HID:
 - a. Tries to obtain the `manufacturer` name, the `model`, and the `serialNumber` of the device. If any of those does not exist, a HID cannot be created.

- b. The three fields are transformed, applied to the URI compliance so they can be used as part of the URI, and concatenated (see Section 5.4.1).
 - c. The result is validated, ensuring no device exists with the same HID.
Note that DDI previously cleaned the manufacturer name, the model, and the serial number, removing bad text such as “no-manufacturer-name”, “serial-number”, “11111111”, or “s/n” that represents noise.
2. If a HID is created, the process is finished. Otherwise:
- If the device is a component and has a `parent`, DeviceHub tries to get a *similar component*: a component of the same `parent` that has no HID and has similar characteristics. For example, if the device is a graphic card (which we call X) from the computer Y, and the graphic card cannot generate a HID, DeviceHub tries to find a graphic card with the same characteristics than X inside the computer Y.
 - If the device does not have a `parent` field, the user is required to: (a) provide an internal identifier of an existing device, so the match can be done, or (b) state that the device is new, so it will be assigned with a new synthetic identifier. This behaviour is configured by the user, and by default the synthetic identifier is an incremental sequence of natural numbers (similar to the *autoincrement* of many SQL databases).

When introducing devices to the system, they generate an ETag³² with the union of the immutable characteristics. The ETag is used to compare if two devices *seem equal*.

Register takes the following considerations:

- The majority of devices that cannot generate HID are components, and it is not crucial to guarantee veracity (system can wrongly match two components that are not the same, or otherwise). Users will not want to spend any extra time to ensure veracity for components. As such, this process needs to be as automated and unattended as much as possible.
- Devices that are not components (Computers, etc.), meaning they cannot have a `parent` field, are more important for the user to ensure veracity, as they usually need to be reported to clients, government, and auditors. In this case, the system tries to generate a HID (which will work in a majority of situations) and otherwise leaves the final decision to the user, as explained above. Note that, as devices may have stuck printed tags with an easy synthetic identifier, users can easily introduce it to the system. Please also note that this process is not entirely accurate, as users can make mistakes.
- We can avoid the user to introduce an identifier for those devices that neither have a parent field and neither generate a HID. One way is valuing if two devices and their components *seem equal*, this is, if we can match their ETags. Another way is using a

³² Mimics the idea of an HTTP ETag (entity tag), which “is a unique identifier for a particular representation (bag of bytes) of a resource” [66], like a hash.

system with more knowledge and authority to set a unique identifier for the device. These mechanisms can be future work.

6.2.4 Managing users

DeviceHub provides an authentication and permission system to manage the users. `Account` is the class that represents a user (see Section 5.2.2.3.2.). This class contains, among others, an email and a password.

6.2.4.1 Authentication

DeviceHubClient provides an interface to authenticate the user, and DeviceHub has an API endpoint `POST /login`, which returns a token to be used in the following operations, using the REST standard. The process is as follows: (1) the user performs `POST /login`, maybe indirectly through DeviceHubClient (see Figure 41); (2) DeviceHub looks for the account in the database; and (3) if it can successfully authenticate the user, it returns the token alongside other information, such as to which databases the user has access.

DeviceHub can offer different databases. In such cases, the `POST /login` method would have returned a list of `databases` and the `defaultDatabase`, which is a value from the list that specifies the database that the client should start using. Every future request to the API will need to specify which database to work with by prefixing the `database` at the beginning of the path in the URI. For example:

Listing 10 Two real examples of HTTP requests with a prefixed database.

```
HTTP GET www.devicetag.io/reutilitza-upc/events/4
HTTP POST www.devicetag.io/public/devices
```

DeviceHubClient offers a commodity switcher for the user to change the active database (see Figure 42).



Figure 41 (Left) The interface of DeviceTag.io (DeviceHubClient) for performing login is minimalist and simple.

Figure 42 (Right) DeviceHubClient offers at the right-top of the screen a button reminding the user which database it is using, and by pressing it a drop-down menu appears with the different databases to which the user has access.

6.2.4.2 The permission system

DeviceHub offers a role based permission system. Every account has a role attached that sets the permission level it has, as well as what the user can do. Roles are ordered hierarchically, so a role above another has more access.

We present the roles in an ascendant order of privilege. This means the first role has less access than the following one.

1. Basic: most basic user. Cannot do anything (except its account). Useful for extraneous users such as auditors or consultants.
2. Amateur: can create devices; however, you can only see and edit the ones it created. The events they can perform are restricted. Amateurs are usually extraneous users that possess some device, and they are interested in just working with that device, isolated from the rest of the system. For example, Reutilitza.cat receivers are considered *amateurs*.
3. Employee: technicians and other people working in the IT Asset Management. They can perform the full spectre of operations and interact with the devices of others.
4. Admin: they can manage other users (except superusers) and they are not restricted in any operation.
5. Superuser: invisible users that can manage administrators and perform development operations. This role is thought for technicians in charge of maintaining and developing DeviceHub.

Permissions can be set per method and endpoint (for example, limiting who can POST a concrete event) or per field (only an administrator can change the `block` field of an account, which blocks or unblocks a user).

The prototype used in Reutilitza.cat has an advanced permission system that let set permissions for individual users to certain places. For example, *the user X, Y, and Z can register devices in the warehouse number 3*. Future work should implement this system in DeviceHub.

6.2.5 Performing events

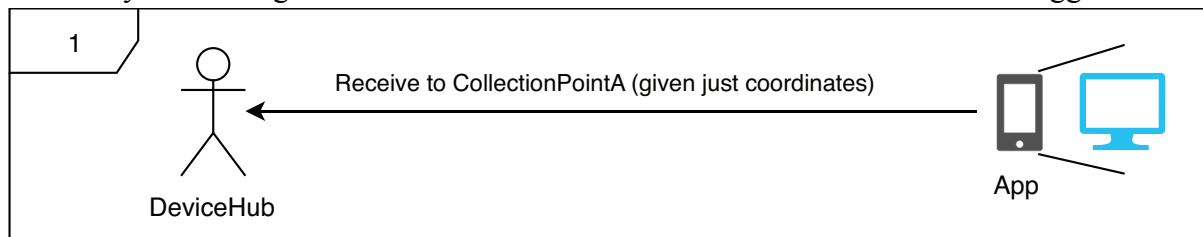
Events are the actions that happen to devices. Events in DeviceHub are described in Section 5.2.2.3.2.

6.2.5.1 Implementation of the collection

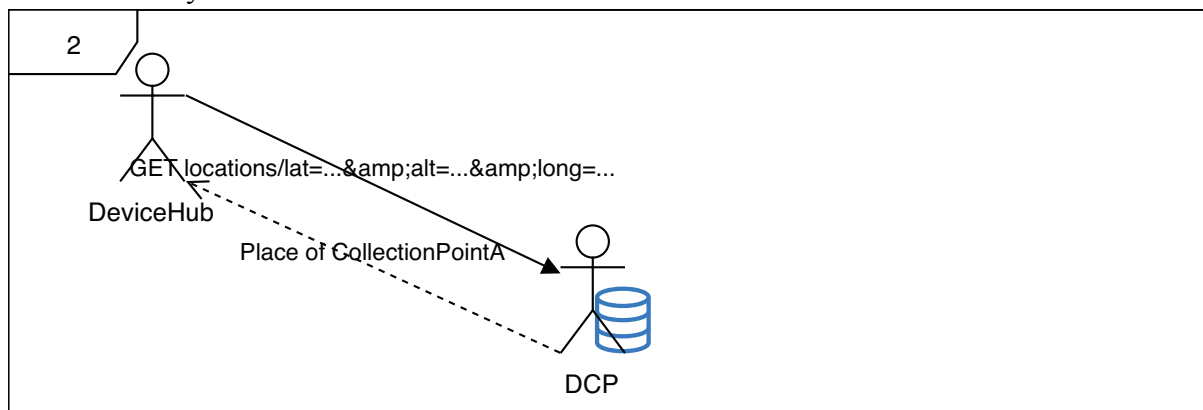
This section explains the implementation of the collection, as explained in Section 5.3.3. It is an example of how DeviceHub collaborates with other agents to complete a process. Note that, at this moment, the implementation is not finished yet.

The collection starts when the device is set to be disposed to a collection point, and it finishes when there is confirmation of the reception. There are many events that interact in this process (and can vary because of the needs of the organization) and, in this case, we mimic the usage in Reutilitza.cat (a social platform where the receiver uses the eReuse.org app). We presuppose that no relevant event has been triggered before.

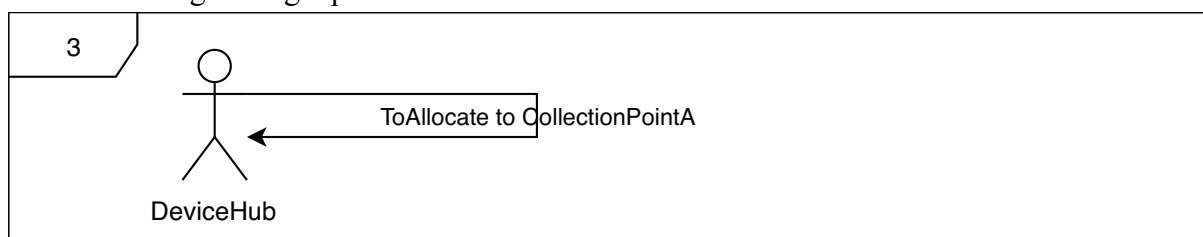
1. The user performs **Receive** to the *CollectionPointA*. If this is done by the eReuse.org app, it only sends the geo-coordinates of the user to the DeviceHub where it has logged in.



2. In order to know to where to perform the event, DeviceHub needs a registered place with an area that englobes the coordinates sent in 1. If the DeviceHub does not have this information, it can mean that the place is managed by another ITAMS. DeviceHub asks DCP about the place whose area englobes the coordinates. Upon success, DCP returns information about the place, such as the type of collection point and the URI of the ITAMS that owns the place. Upon failure, depending on the policies of the DeviceHub, the user is asked to create a new place, which can be validated later by the administrator or the community.

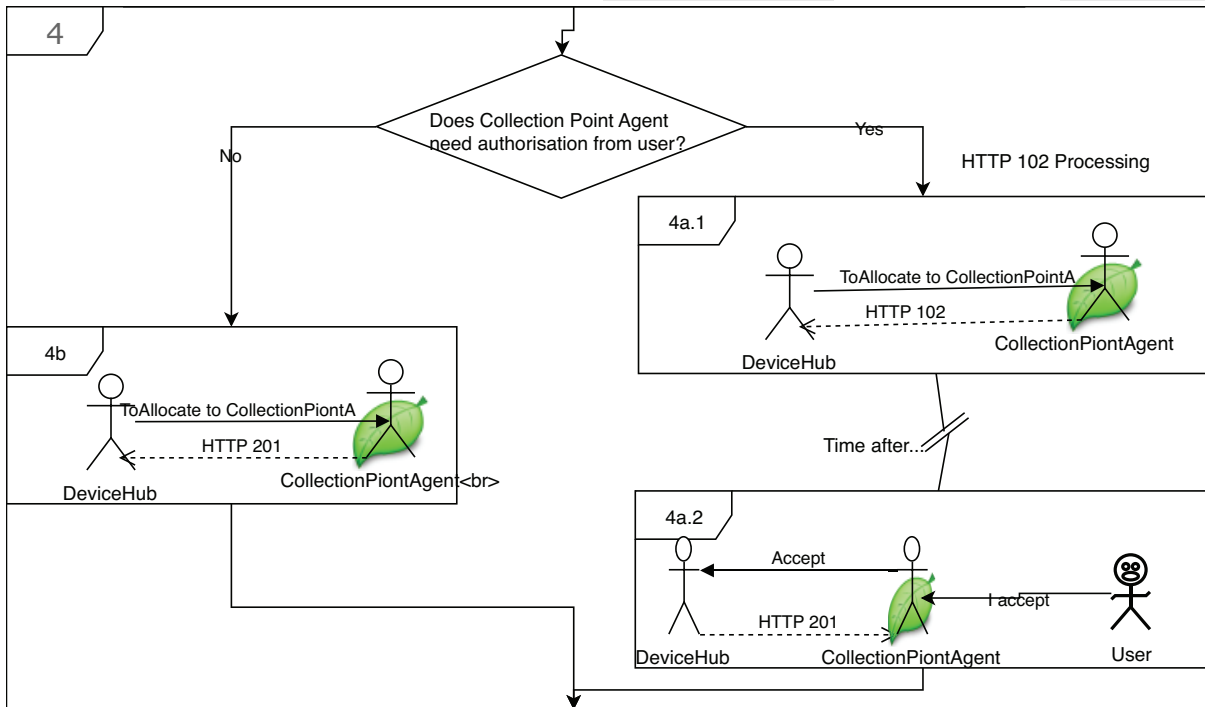


3. If no one **Allocated** (legal transferal) the device to the *CollectionPointA* prior to performing a **Receive** (physical transfer), DeviceHub needs to **try** (**toAllocate**) transferring the legal possession of the device.

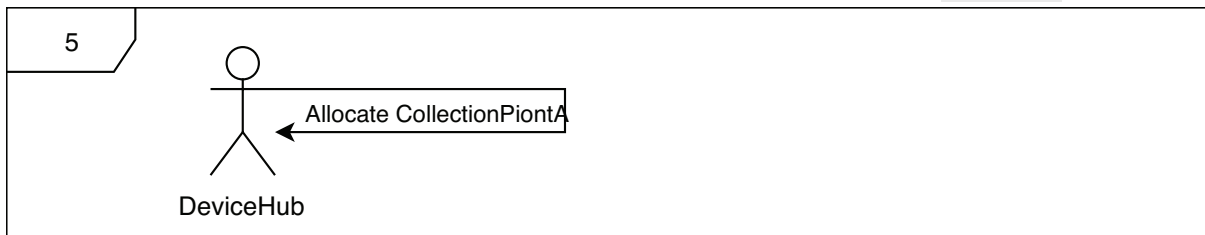


4. At this moment, two things can happen:
 - a. The ITAMS managing the collection point needs for a user to authorize the collection. This user can be an employee working in the collection point, for example. In this case, **toAllocate** is done (4a.1) and it is answered by **HTTP 102 Processing**, meaning that the request has been successfully received, but there is no real answer yet. After some time, the user accepts the **toAllocate** from its ITAMS, and the *CollectionPointAgent* connects to DeviceHub to perform **Accept**.

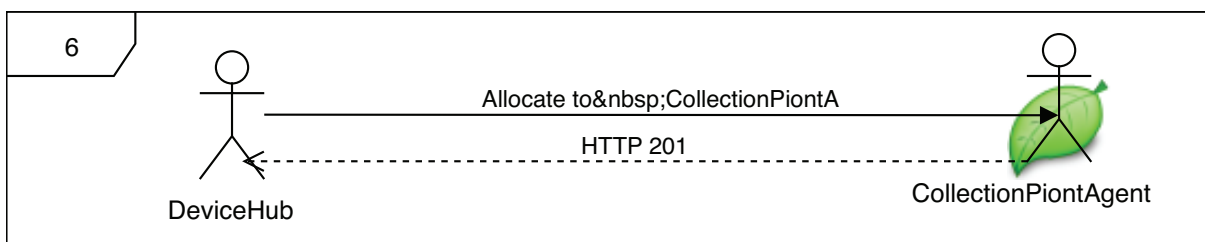
- b. The ITAMS managing the collection point does not need authorization from a user (maybe it has some built-in criteria, or it always accepts or refuses the allocation). In this case, the ITAMS answers a HTTP 201 Created when receiving a toAllocate.



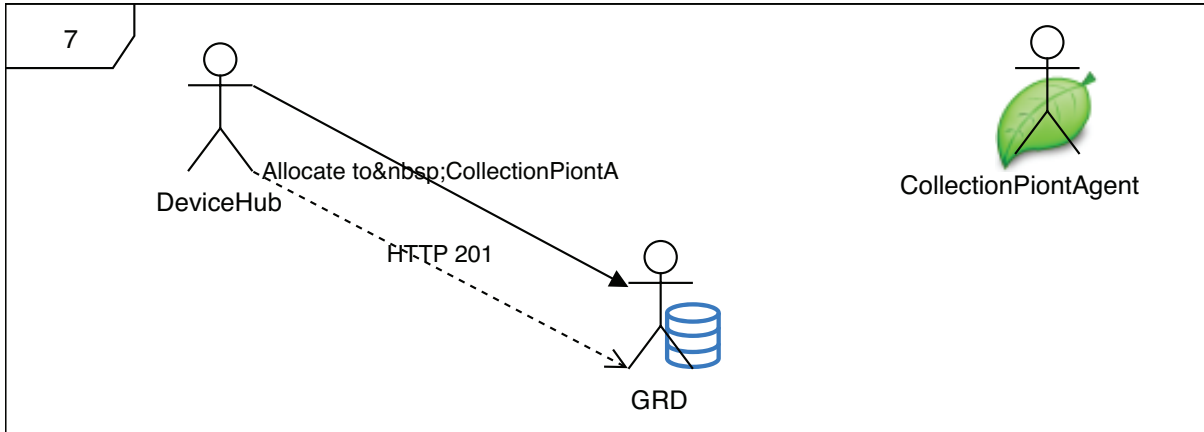
5. Once the willingness for allocation is resolved, DeviceHub performs Allocate.



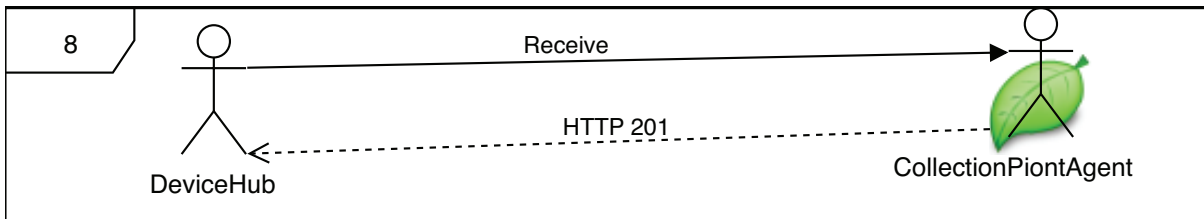
6. Allocate is sent to the CollectionPointAgent, which acknowledges the possession of the device.



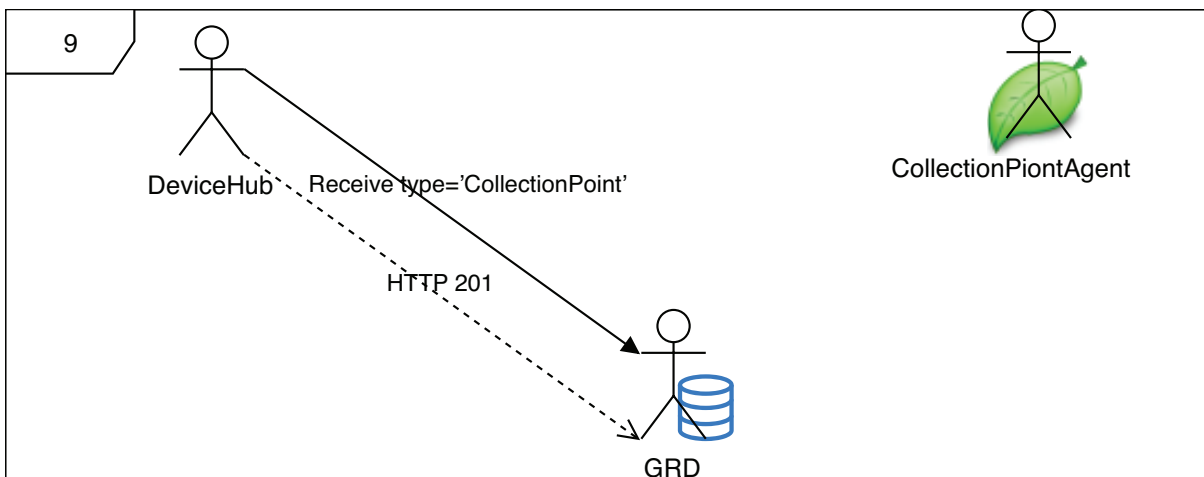
7. DeviceHub sends the event to GRD, as Allocate is an events GRD accepts. Note that DeviceHub is still the holder of the device, so GRD does not tolerate any communication from CollectionPointAgent.



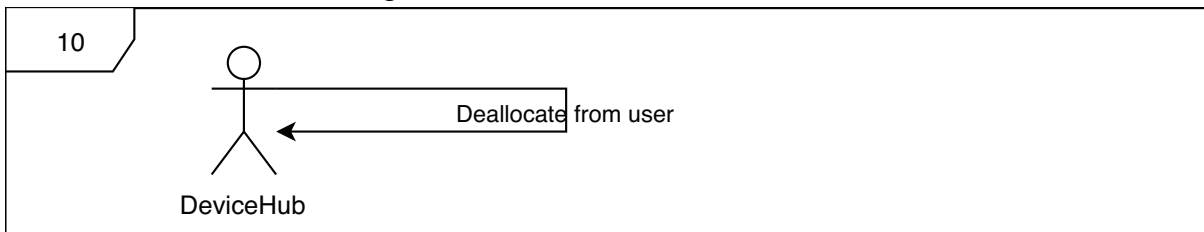
8. After the allocation, the Receive itself is performed to CollectionPointAgent.



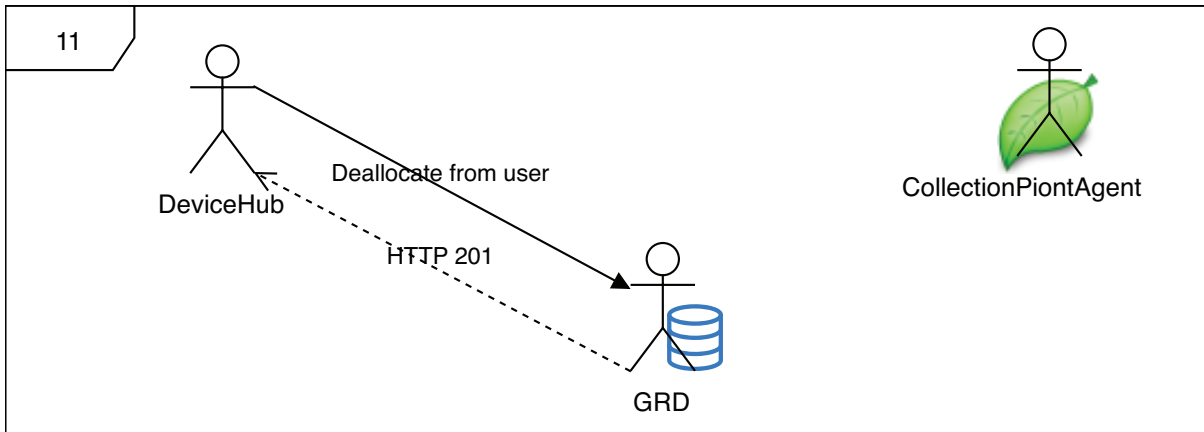
9. And to GRD.



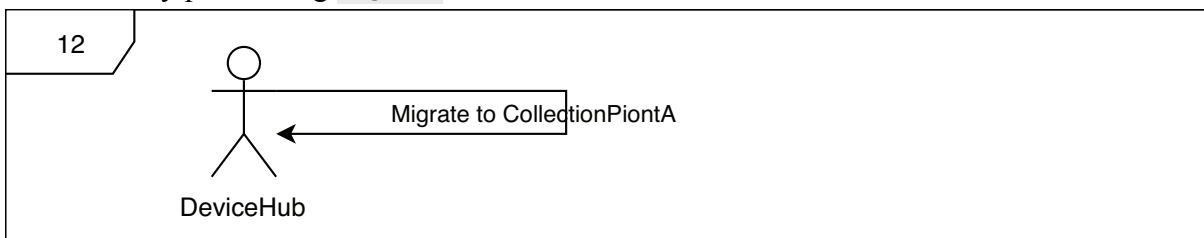
10. Disposing a device to a collection point means transferring the possession; the user loses the legal right of the device. Deallocate is performed. Note that this event is not interesting for the CollectionPointsAgent.



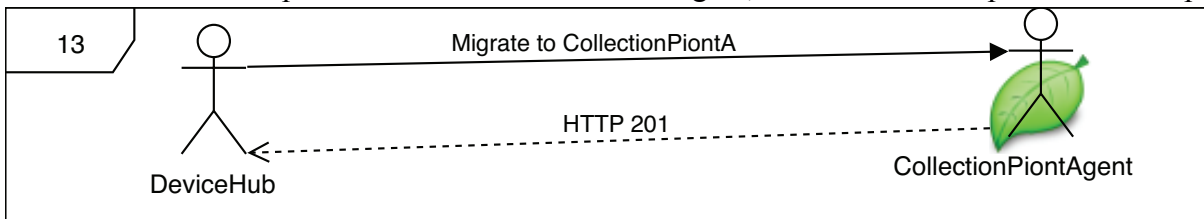
11. Deallocate is sent to GRD.



12. Finally, to finish the transferal, we need to change the ITAMS holder (in charge) of the device by performing **Migrate**.



13. The event will be performed to **CollectionPointsAgent**, where it will accept the ownership.



14. In the end, DeviceHub sends **Migrate** to GRD. As this event specifies, DeviceHub will not send more events to this device, and the only ITAMS that will be able to send them to GRD is **CollectionPointAgent**. If another system sends an event regarding this device, it will be considered an exception, and **CollectionPointAgent** will be notified. This is the intended behaviour to warn IT asset managers when finding lost devices (which can be contaminating in a waste dump or a river, for example).

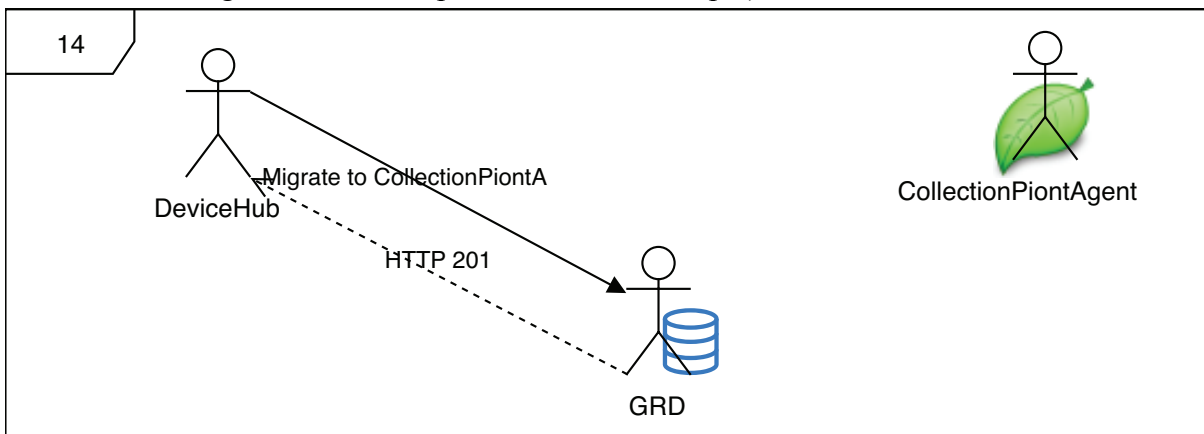



Figure 43 Representation of the workflow of a collection.

6.2.6 Managing places

DeviceHub offers CRUD of places (Create, Read, Update, and Delete), and users can use the API, DeviceHubClient, or the eReuse.org App to create and manage them. When creating a place, you can set a geographical area and then perform events inside those areas using the GPS capabilities of smartphones, as shown in the previous section.

When using the prototype, we found that one of the most used functionalities with places is to know which devices are in there and which events were performed inside. This is why DeviceHub offers a way to filter devices and events by place name, and DeviceHubClient uses a list of places as part of its three-column design; when the user selects one of the places on the list, only devices in that place are shown.

Moving devices from place to place was the second most common operation regarding devices, and this has taken special care in DeviceHub: a client can use PATCH /places setting `devices` and DeviceHub will automatically update the devices in that place, removing the ones that do not appear in the request. In DeviceHubClient, this action can be performed by two clicks: first, by selecting the device, and then pressing  in the place.

Finally, DeviceHub exchanges information about public and interesting places from the point of view of reusing and recycling to DCP, and this is the intended way to feed DCP. In the implementation of the collection, DCP had information about CollectionPointA because CollectionPointsAgent created it and shared it with DCP. Actually, it is done manually, so this exchange is not implemented yet and should be done in future work.

6.2.7 Tagging devices

Tagging a device is one of the most important aspects in reusing, as it increasingly accelerates performing any kind of event on devices, specifically when using the eReuse.org App. If there were no tags, the user would have to use any other printed tag (such as the serial number of the manufacturer) or executing DDI again and uploading the file to DeviceHub to generate an identifier.

Another way of identifying already-registered devices is by using Radio Frequency Identification (RFID) technologies, which have the following advantages over traditional ways: (1) no line of sight required, as it uses radio frequencies, (2) mass identification in

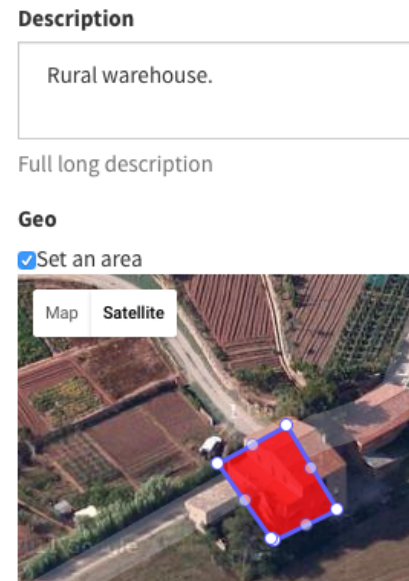


Figure 44 Setting the area of a place in DeviceHubClient, with the assistance of Google Maps.

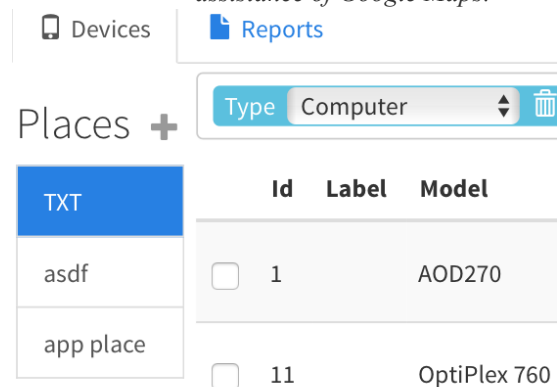


Figure 45 Selecting a place filters the list of devices, showing only the devices that are on that place.

seconds, and (3) they can be mounted in strategic places such as doors to count the devices passing by automatically [53]. A technology such as this can be adopted in the future (and can co-exist with traditional tagging); however, it has the major inconvenience of being more expensive, as the price of a RFID chip is significantly greater than a sticker; the difficulty of tagging only one device, when there are many devices near others; the need of an adoption through all layers of the work chain; and the inability of using smartphones to scan tags, which means investing in machinery. Phones could use NFC technologies to read RFID, but the working distance of NFC is too narrow to be useful for our purposes [54], and not many smartphones have unblocked NFC chips.

Tags generated by DeviceHub are composed of (from top to bottom, see Figure 46): (1) a logo representing the organization that prepared the device for reuse or similar, (2) a QR with the URI of the device, and (3) a list of key value identifiers, at least an easy and synthetic identifier used in the URI by the DeviceHub, and the `labelId`. Note that by having an URI, the QR can be used by any compatible scanner, in which case it usually opens a web browser where the user sees the device through DeviceHubClient³³.



Figure 46 A tag generated by DeviceHub, in concrete by DeviceTag.io.

DeviceHubClient can generate personalized tags for devices. Once the user selects the devices it wants to tag, a *Tag* button appears. Clicking it shows Figure 47, where the user can modify the sizes and margins of the tags, toggle the usage of a logo, change the default logo by uploading a custom image, preview their result in real time, and finally print them. Printing tags actually generates a PDF, so the result can be sent to the one in charge of printing.

³³ The only difference when using a regular web browser or the eReuse.org App is the `HTTP Accept` header: if the server detects an `HTTP Accept` header with `HTML` value (the default for any browser), it will redirect the user to DeviceHubClient. The eReuse.org App however, sends an `HTTP Accept` header with only `application/json` value, so the server returns the JSON representation of the device.

Figure 47 Main options in DeviceHubClient when personalizing the tags.

6.2.8 Interacting through the eReuse.org ecosystem

We have already seen how DeviceHub needs to collaborate with other agents of the eReuse.org system (other DeviceHub, GRD, DCP).

In the case of systems that want to receive certain events when fired, such as GRD or e-commerce websites, DeviceHub offers a core module called *Logger*. This module creates a background thread (a daemon) whose mission is to send the events DeviceHub performs to different registered agents, such as GRD (through HTTPS in this case). The daemon awakens when there is a new event in the queue; it sends it to the respective systems, and then goes to sleep again. This way, DeviceHub can answer the request of the user without having to wait for other systems to receive the event, speeding up the interaction. *Logger* is used by *GRD_Logger*, which adapts the information sent to the schema of GRD.

6.2.9 Emitting certificates

One of the most important values in ITAM is ensuring that the processes they manage are successfully done, such as identification or disk erasures. If the results are wrong, the organization can face a lawsuit. For example, if personal information of a user is not successfully erased from a hard-drive, this data can be recovered and used by a malicious third party, threatening privacy [55]. To ensure those processes, organizations and governments enforce issuing certifications guaranteeing correct completion. To be able to generate a certificate we need to:

- Ensure IT asset identification. This is, to be able to identify the devices we are issuing the certificate to correctly.
- Ensure that the process to generate the certificate has been correctly done. We integrate special mechanisms to DDI to report erasures only when they are 100% done and state otherwise even in cases of even minimal error. These erasures tend to take hours to be valid, and this is the major difference between the events `EraseBasic` and `EraseBySectors`. The former is faster but cannot generate a complying certificate, and the latter is slower but can generate one.
- Ensure that the whole process is not tampered and data cannot be modified. We cannot certify a process where users can introduce or modify sensible data, as they can be mistaken or have bad intentions. DDI can sign the resulting JSON file with a private key created by the user in DeviceHub, and DeviceHub only accepts the file if the signature is correct and is from the same user. If a user is found tampering, all its introduced data and certificates generated are invalidated, and the key is blacklisted. This is a feature in development.
- Optionally, to be certified by public authorities. In the majority of countries, for certifications to have legal value, they need to be certified by a trusted organization.

eReuse.org proposes the community to contribute in getting the public certifications, becoming collective goods or peer property [12].

6.2.10 Extendable and integrable

For DeviceHub to succeed, it needs to be extended and integrated by third parties into their ITAMS and other systems, as it is not the objective of DeviceHub to replace them, but to offer a mechanism to manage and ensure reusing and recycling. We have done many decisions in DeviceHub to accomplish these two premises.

6.2.10.1 *Extendible schema*

Eve, the framework DeviceHub is built on, generates the RESTful API by interpreting Python dictionaries, whose schema is extended from Cerberus. Eve expects a dictionary per REST endpoint, and this dictionary contains information about how Eve needs to behave with the endpoint, and the schema of the data in the endpoint. The system is quite personalized; however, it lacks one important thing used in DeviceHub: class hierarchy. In DeviceHub, we have a few top level endpoints defined (devices, events, accounts, and places) and the rest inherits them. We want third parties to create sub-types of devices, events and places, so we provide an easy mechanism to extend an endpoint and its schema. For example, the endpoint `/devices/components` extends from `/devices`, and `/devices/components/hard-drive` extends from `/devices/components`.

To provide this functionality we offer the Python classes `ResourceSettings` and `RDFS`; both are base abstract classes thought to be respectively extended for endpoints and data schemas, and have a function to generate a resulting dictionary with the information required by Eve to *turn-on* endpoints. The class diagrams in Section 5.2.2.3 are a full representation of RDFS and all classes that extend from it. Users can extend any class of the diagram to add custom behaviour. Some events that have no special logic (`Ready`, `toPrepare...`) are generated programmatically in execution time, and it is a first step to offer, in the future, a way for users to generate and customize events through an interface such as `DeviceHubClient`.

6.2.10.2 *Extensions*

Extensions (or plug-ins) add extra functionality to DeviceHub that is not suitable to be in the *core*. Candidates for extensions are the code to integrate DeviceHub with other private systems, and custom specific functionality.

At the moment of this writing, we are preparing DeviceHub to offer extensions, by using the mechanisms and directives implemented in Flask [56], the micro-framework Eve is built on. Flask already has many extensions built [57], so using the same mechanisms we ease the extension development (see in <http://flask.pocoo.org/docs/0.10/extensiondev/> for how to develop a Flask extension). We made *devicehub-doc* [58] as an exemplifying basic extension using the extendible schema. It generates API and schema-related documentation from a DeviceHub.

6.2.10.2.1 Automatically configuring DeviceHubClient through the schema

One of the most notorious mechanisms to extend and integrate DeviceHub is the usage of the `/schema` endpoint³⁴, which exposes the schemas of every endpoint to which the user has access. The information that DeviceHub shares for every field in the schema is rich, the API section in the DeviceHub documentation (in the additional material) is, in part, the information the schema exposes, and is as follows:

- The `name` of the field.
- The `type` of the field.
- If the field is a reference (`data_relation`) to another resource. For example, the field `device` in an `Event` is a reference to the resource `Device`.
- If the value of the field needs to be `unique` among the same resource.
- If there is a `default` value, in case the client does not send it.
- If the value is `required`.
- A human `description` for the value.
- If the value is `write-only`. This means that the value can be written, but not read (GET will not show the value).
- If the value is `read-only`. This means that the value can only be read.
- If the value is `modifiable`. This is, it can be updated once set (PATCH or PUT cannot submit the value once this has been set before).
- The `unitCode` of the value: the unit of measurement, using the UN/CEFACT Common Code (three characters), were the field must be in.
- Technical `documentation` about the field.
- How much the field `sinks`, or the recommended order for presenting the field. Higher values take the field above in a list, and lower values makes it *sink*, going down to the list.
- Roles with writing permission and roles with reading permission.
- `OR`, a key/name map referencing if at least one of many fields must be present.
- If one field `excludes` another, this is so that a field cannot be present at the same time as another one.

By obtaining this information, clients can automatically understand and generate appropriate schemas. For example, when adding a new type of event with some new fields to DeviceHub, there is no need to update the code of the client, as it automatically updates its own schema by interpreting `/schema`. In DeviceHubClient it is done after the user logs in. The most visual example is configuring the creation and editing forms for the events and places (see Figure 48). DeviceHubClient translates the structure of the schema endpoint to Angular Formly, an open-source module that creates HTML forms programmatically. To avoid a costly request to the schema every time the user refreshes a client, both DeviceHub and DeviceHubClient honour the HTTP Cache. This way, it is easier to extend DeviceHub, as many configurations are automatically handled by the clients, so there is no need to re-program them.

³⁴ Learn more from the schema endpoint here: <http://python-eve.org/features.html#the-schema-endpoint>

Figure 48 DeviceHubClient generates the forms by translating the information in the schema endpoint to the specification of Angular Formly, and extends this module by providing functionality. In this case, the DeviceHubClient understands that a place needs a list of devices, so it offers a widget to the user to select some devices. It also understands that a place needs a pair of geolocation coordinates, so it adds a widget of Google Maps for the user to select a position.

At the moment of writing this document, there are alternatives that offer the same idea (RESTful API schemas with auto-configurable clients), such as OpenAPI³⁵, RAML³⁶, api blueprint³⁷ and Hydra³⁸. While expecting the projects to mature and a candidate to stand out, we use the schema of Eve, extending it with Hydra when necessary (which is backed up by the W3C).

6.3 API

The main way to communicate with DeviceHub is through the API. The API is RESTful using HATEOAS directives, partially extends Schema.org and uses a limited subset of JSON-LD. Although the generic aspects of the API (shared with other eReuse.org systems) are described in Section 5.2.4, this section details specific ones. See the API section in the DeviceHub documentation from the additional material for the definition of each endpoint.

³⁵See it at <https://openapis.org>.

³⁶See it at <http://raml.org>.

³⁷See it at <https://apiblueprint.org>.

³⁸See it at <http://www.hydra-cg.com>.

Built on Eve, you can filter³⁹ (Listing 11), order⁴⁰ (Listing 12), embed⁴¹ (Listing 13) and project⁴² (Listing 14) the results of any HTTP GET request. Embedding one field means replacing the reference to an object by the actual contents of such object.

Listing 11 Exemplifying GET query filtering results. In this case, it obtains all events from the public database of DeviceTag.io that are snapshots. It is using MongoDB syntax, so it supports the majority of filters MongoDB uses.

```
https://api.devicetag.io/public/events?where={"@type": "Snapshot"}
```

Listing 12 Exemplifying GET query ordering results. In this case, it obtains all devices from the public database of DeviceTag.io, sorting them by the fields of byUser (ascending) and byOrganization (descending). It is using native Python syntax.

```
https://api.devicetag.io/public/devices?sort=_created,-type
```

Listing 13 Exemplifying GET query embedding results. In this case, it obtains all devices from the public database of DeviceTag.io, and embeds the user object, and a list with all the objects of all events, using MongoDB syntax.

```
https://api.devicetag.io/public/devices?embedded={"byUser":1, "events":1}
```

Listing 14 Exemplifying GET query with projection definition. In this case, it obtains all places from the public database of DeviceTag.io; however, it does not retrieve the field byUser. It uses MongoDB syntax.

```
https://api.devicetag.io/public/places?projection={"byUser":0}
```

Listing 15 Exemplifying GET query combining different operations. In this case, it obtains all devices from the public database of DeviceTag.io that are TFT (sub type of monitors) and it sorts them by labelId (ascending).

```
https://api.devicetag.io/public/devices?where={"type": "TFT"}&sort=[("labelId",1)]
```

6.3.1 Login

To use the API, you will need first to log in with an existing account from the DeviceHub. Perform POST /login with the email and password fields filled (see Listing 16). Upon success, you will be answered with the account object (see Listing 17), containing a Token field. From this moment, any other following operation against the API will require the following HTTP Header: Authorization: Basic token.

Listing 16 Exemplifying a login request.

```
POST /login
Content-Type: application/json
{
  "email": "example@example.com",
  "password": "example"
}
```

Listing 17 Successful login response.

```
{
  "databases": [
    "example_database",
    "example_database_2"
  ]
}
```

³⁹More information at <http://python-eve.org/features.html#filtering>.

⁴⁰More information at <http://python-eve.org/features.html#sorting>.

⁴¹More information at <http://python-eve.org/features.html#embedded-resource-serialization>.

⁴²More information at <http://python-eve.org/features.html#projections>.

```
],  
"defaultDatabase": "example_database",  
"password": "sha256 codified password",  
"role": "admin",  
"email": "example@example.com",  
"_id": "149fja02umz1",  
"@type": "Account",  
"token": "Base 64 codified token"  
}
```

6.3.2 Working with devices

There is no `POST /devices`. To upload information of devices with tests, erasures, etcetera, use `POST /snapshot`, and to only create a device use `POST /register`; they will take care of everything.

7 SUSTAINABILITY

In this section we describe the environmental, social and economic impact of the project.

7.1 SOCIAL AND ENVIRONMENTAL DIMENSIONS

Reusing prevents generating e-waste. E-waste is the fastest growing type of waste, particularly in some developing countries where the volume is expected to grow by up to 500 times over the next decade, and an estimated 80% of it is still going into landfills and incinerators [59]. Pilots performed with the Catalan Government show that in public institutions, more than 80% of the digital devices to be recycled are operational.

Companies can internally reuse more easily and cheaply, reducing their ecological footprint.

We help to generate open data about traceability, recycling locations, and the durability of devices. This enables building the knowledge base for environmental action, for moving toward a fully circular electronics economy, and reducing leakage. Leakage mainly affects developing countries, where devices are poorly collected and treated in informal waste dumps. 59% of waste is not collected [60]. Recycling operations, primarily managed by the informal sector, use inappropriate techniques that severely jeopardise the health of workers and cause major environmental harm.

When receivers of devices are organizations with social projects, reusing strengthens the digital skills of its stakeholders and helps reduce the digital divide. Finally, empowering social projects helps in promoting positive social change. Our pilots in Catalonia show there is a local demand from social institutions for devices without any upgrade or repair needed.

7.2 ECONOMICAL DIMENSION

This project helps in increasing efficiency and reducing costs for professional companies and organizations that reuse and recycle. Ultimately, it boosts the economy, creates new jobs and improves companies' competitive position. Reuse has the potential to employ 10 times more people per ton of material processed than recycling activities. The job potential in repairing activities is huge. It is a labour-intensive service that requires both manpower and skills, which is difficult to send overseas; therefore, countries generating WEEE stand to benefit. As an example, in Ghana, where 80% of devices are not recycled but refurbished and re-used [61], there are 1,200 repairers, which generate income for more than 30,000 people [62]. On the other end, on average, 50% of people in Europe would be happy to buy a second-hand appliance, which are 30-40 percent cheaper [63]. Finally, for some social communities, as for our pilots, the usage of the eReuse.org tools are the difference between being viable or not, as they can reduce costs and increase revenue enough to be able to hire people (instead of just basing on voluntarism).

The economic viability of the eReuse.org project depends on a periodic payment for the offered tools and services, plus a variable fee, which depends on the number of successfully reused devices. The eReuse.org project still needs to set these amounts. On the other side, the

cost of the project is kept low as the tasks that are going to be developed are the strictly necessary to have a minimum viable product, which is going to be used to start generating income in order to develop more functionality.

7.3 SUSTAINABILITY MATRIX

Taking into consideration everything we said in this section and in other ones, we can value the sustainability of our project using the matrix presented by Christian Felber in “The economy of the common good”. Briefly, the main objective of the project is to empower people with software to bootstrap reuse, which helps reducing WEEE and promote social change when reuse is done to social projects. The project is co-designed with stakeholders, so there have been changes and additions that have made the project costlier. However, thanks to the methodology used, Scrum, and an initial versatile planning, adaptation has been easy, and cost and time have not affected drastically.

Table 6 Sustainability Matrix, extracted from Christian Felber “The economy of the common good” [64].

Sustainable?	Economic	Social	Environmental
Planning	Economic viability	Improved quality of life	Resource analysis
Assessment	7	9	10
Results	Final cost vs previewed	Impact on social environment	Resource consumption
Assessment	5	9	8
Risks	Adaptation to changes	Social damage	Environmental damage
Assessment	-5	0	0
Total	43		

8 CONCLUSIONS

Digital devices are mass produced in a world of finite resources, and they have become a requirement for people to participate in society. Creating a circular life cycle of devices is important if we want to ensure our lifestyle, greatly reduce our ecological footprint, spur social change by reducing the digital divide, support social projects, and ultimately boost our economy by creating new jobs and improving the competitive position of industry. To reach circularity, there is a need for specific software, traceability systems, and standards.

This project defines the open-source, decentralised, local, and scalable eReuse.org architecture, which is a framework for governments, organizations, and individuals to reuse digital devices by: defining a common standard ontology and API to manage the preparation for reuse, transferal, and collection; ensuring reusing and recycling by defining the Hardware Identifier (HID), acknowledging final usage, and specifying a global traceability system; integrating the eReuse.org system with organizations; maximizing the social usage of devices; and generating open-data to measure circularity. Moreover, this project develops DeviceHub, an open-source IT Asset Management System (ITAMS) focused on efficiently and securely managing reuse until collection for recycle, with the objective of enabling sustainable collaborative reusing. DeviceHub consists of two technical projects: the DeviceHub itself (a REST server) and DeviceHubClient (a browser web client). In this documentation, we detail the design: how it manages the metadata of devices, users, places and events, and how it creates tags, collaborates with other agents, emits certificates, and supports auto-configurable extensions.

One of the key contributions of this project has been reflected in achieving the Best Paper award at Environinfo 2015 (University of Copenhagen, Denmark) for *Breaking Barriers on Reuse of Digital Devices Ensuring Final Recycling* [65]. On a personal note, I have learned many new technologies such as Flask, Eve, Angular, and MongoDB, and concepts such as linked data and ITAM. I am still learning managing requirements, deadlines, schedules, and problems. However, my richest experience has been travelling and collaborating, not only in conferences, but with associations and communities: from big and important to small and remote, I have always learnt more than the project I was bringing to them.

Future work is standardizing the ontology, improving the software, replicating it in more countries, and welcoming stakeholders in the co-design and co-development. The eReuse.org software is a work-in-progress, and this project is one milestone in a long trip towards a full circular economy, where our electronic devices do not harm the environment and people as much anymore, but instead help them in promoting social change.

9 BIBLIOGRAPHY

- [1] D. Franquesa, L. Navarro, D. López, X. Bustamante and S. Lamora, “Breaking Barriers on Reuse of Digital Devices,” p. 8, 2015.
- [2] A. Cerrillo, “Residus electrònics fora de control,” *La Vanguardia*, pp. 28-29, 29 05 2014.
- [3] J. Huisman, I. Botezatu, L. Herreras, M. Liddane, J. Hintsa, L. d. Cortemiglia, V., P. Leroy, E. Vermeersch, S. Mohanty, S. van den Brink, Ghenciu, B., Dimitrova, D., Nash, E., Shryane, T., Wieting, M., J. Kehoe, C. Baldé, F. Magalini, A. Zanasi, F. Ruini and A. and Bonzio, “Countering WEEE Illegal Trade (CWIT) Summary Report, Market Assessment, Legal Analysis, Crime Analysis and Recommendations Roadmap,” CWIT, France, 2015.
- [4] C. Baldé, F. Wang, R. Kuehr and J. Huisman, *The global e-waste monitor -2014*, Bonn: United Nations University, IAS - SCYCLE, 2015.
- [5] United Nations, “Trends in sustainable development, Chemicals, mining, transport and waste management,” April 2010. [Online]. Available: https://sustainabledevelopment.un.org/content/documents/28Trends_chem_mining_transp_waste.pdf. [Accessed 12 May 2016].
- [6] Proparco, “Private Sector & Development,” *Proparco's Magazine*, no. 15, October 2012.
- [7] I. T. U. (ITU), “Press Release: ITU releases latest tech figures & global rankings,” 7 10 2013. [Online]. Available: http://www.itu.int/net/pressoffice/press_releases/2013/41.aspx#.VhOJfDahd9B. [Accessed 10 2015].
- [8] International Telecommunication Union (ITU), “Internet user per 100 inhabitants, 1994-2007,” 2007. [Online]. Available: <http://www.itu.int/ITU-D/ict/statistics/ict/graphs/internet.jpg>. [Accessed 10 May 2016].
- [9] D. Franquesa, J. Cruz, C. Alvarez, F. Sánchez, A. Fernández and D. López, “The social and environmental impact of engineering solutions: from the lab to the real world,” *Int J Eng Educ*, vol. 26, no. 5, pp. 1144-1155, 2010.
- [10] Ellen Macarthur Foundation, “Circularity indicators. An approach to Measuring Circularity. Project Overview,” May 2015. [Online]. Available:

- <http://www.ellenmacarthurfoundation.org/circular-economy/metrics>. [Accessed 23 September 2015].
- [11] eReuse.org, “Project Definition,” eReuse.org, [Online]. Available: <https://wiki.ereuse.org/def:start>. [Accessed 10 May 2016].
- [12] D. Franquesa, L. Navarro and X. Bustamante, “A Circular Commons for Digital Devices. Tools and Services in eReuse.org,” in *ACM LIMITS*, Irvine, 2016.
- [13] Gartner, “Gartner IT Glossary, IT Asset Management (ITAM),” [Online]. Available: <http://www.gartner.com/it-glossary/it-asset-management-itam>. [Accessed 23 09 2015].
- [14] International Association of Information Technology Asset Managers, Inc., “IAITAM,” 2014. [Online]. Available: <http://iaitam.org>. [Accessed 13 November 2015].
- [15] Reutilitza.cat, “Reutilitza.cat,” [Online]. Available: <https://www.reutilitza.cat>. [Accessed 09 2015].
- [16] Independent, “GLPI,” 2003. [Online]. Available: <http://www.indepnet.net/>. [Accessed 9 2015].
- [17] GLPI, “GLPI,” [Online]. Available: <http://www.glpi-project.org/spip.php?lang=en>. [Accessed 9 2015].
- [18] GLPI, “Characteristics,” [Online]. Available: <http://glpi-project.org/spip.php?article53>. [Accessed 12 May 2016].
- [19] GLPI, “Screenshots of GLPI 0.7,” [Online]. Available: <http://glpi-project.org/spip.php?article42>. [Accessed 12 May 2016].
- [20] Allegro Group, “Ralph,” [Online]. Available: <http://allegro.tech/ralph/>. [Accessed 9 2015].
- [21] OCS Inventory Development Team, “OCS Inventory,” [Online]. Available: <http://www.ocsinventory-ng.org/>. [Accessed 9 2015].
- [22] SpiceWorks Inc, “SpiceWorks,” [Online]. Available: <http://www.spiceworks.com/>. [Accessed 9 2015].
- [23] Landesk Software, “About Landesk,” [Online]. Available: <http://www.landesk.com/company/about/>. [Accessed 9 2015].

- [24] Reutilitza.cat, “Reutilitza.cat,” 21 December 2010. [Online]. Available: <https://www.reutilitza.cat>. [Accessed 4 October 2015].
- [25] F. Ramírez Lázaro, “Creació de les eines necessàries per a la reutilització,” Barcelona, 2012.
- [26] S. Caballero Peña, “Plataforma web per a la reutilització d’equips informàtics,” Barcelona, 2012.
- [27] J. A. Giner, “Plataforma Web 2.0, Xarxa de Suport a la Reutilització TIC,” p. 124, 9 2011.
- [28] Hojtsy, Gábor, “Drupal 6.0 released,” 13 2 2008. [Online]. Available: <https://www.drupal.org/drupal-6.0>. [Accessed 9 2015].
- [29] Drupal.org, “Drupal 8 development cycle,” 11 09 2015. [Online]. Available: <https://www.drupal.org/core/dev-cycle>. [Accessed 23 09 2015].
- [30] D. Buytaert, “Why the big architectural changes in Drupal 8,” 9 9 2013. [Online]. Available: <http://buytaert.net/why-the-big-architectural-changes-in-drupal-8>. [Accessed 9 2015].
- [31] A. Álvarez, R. de las Heras and C. Lasa, *Métodos Ágiles y Scrum*, Madrid: Ediciones Anaya Multimedia, 2012.
- [32] Wrike, Inc., “Wrike,” 2006. [Online]. Available: <https://www.wrike.com>.
- [33] Google, “Dashboards: Android Developers,” 4 April 2016. [Online]. Available: <http://developer.android.com/intl/es/about/dashboards/index.html>. [Accessed 16 April 2016].
- [34] Schema.org, “Schema.org,” [Online]. Available: <https://schema.org/docs/about.html>.
- [35] H. Knublauch, “OWL version of schema.org,” 18 December 2015. [Online]. Available: <http://topbraid.org/schema/>. [Accessed 20 December 2015].
- [36] R. V. Guha and D. Brickley, “Schema.org: evolution of structured data on the web,” *Communications of the ACM*, vol. 59, no. 2, p. 47, February 2016.
- [37] Automotive ontology community group, “Automotive ontology community group,” [Online]. Available: <https://www.w3.org/community/gao/>. [Accessed 15 October 2015].

- [38] Bib.schema.org, 28 April 2015. [Online]. Available: <https://www.w3.org/community/schemabibex/wiki/Bib.schema.org-1.0>. [Accessed 1 May 2016].
- [39] LOV, “Linked Open Vocabularies,” [Online]. Available: <http://lov.okfn.org/dataset/lov/>. [Accessed 15 June 2015].
- [40] Schema.org, “Schema.org,” [Online]. Available: <https://schema.org/docs/documents.html>. [Accessed 15 July 2015].
- [41] M. Hepp, “The Product Types Ontology: High-precision identifiers for product types based on Wikipedia,” [Online]. Available: <http://www.productontology.org>. [Accessed 12 May 2016].
- [42] P. A. Laplante, Dictionary of Computer Science, Engineering and Technology, CRC Press, 2000, p. 366.
- [43] WHATWG, “URL Standard,” 14 April 2016. [Online]. Available: <https://url.spec.whatwg.org>. [Accessed 21 April 2016].
- [44] R. T. Fielding and R. N. Taylor, “Principled design of the modern Web architecture,” in *ICSE '00 Proceedings of the 22nd international conference on Software*, New York, 2000.
- [45] D. Crockford, “JSON: The fat-free alternative to XML,” *Proc. of XML*, vol. 2006, 2006.
- [46] W3C, “JSON-LD,” 16 January 2014. [Online]. Available: <https://www.w3.org/TR/json-ld/>.
- [47] W3C, “Hydra Core Vocabulary,” 20 March 2016. [Online]. Available: <http://www.hydra-cg.com/spec/latest/core/#description-of-http-status-codes-and-errors>.
- [48] Waste & Resources Action Programme, “PAS 1414 Protocol: Product Guide,” July 2013. [Online]. Available: <http://www.wrap.org.uk/sites/files/wrap/PAS%20141%20Product%20Guide%20-%20Desktop%20and%20laptop%20computers.pdf>. [Accessed 8 April 2016].
- [49] J. Wunder, A. Halbardier and D. Waltermire, “Specification for Asset Identification 1.1,” National Institute of Standards and Technology, Gaithersburg, 2011.
- [50] S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey and K. Beznosov, “What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of

- OpenID,” in *SOUPS '11 Proceedings of the Seventh Symposium on Usable Privacy and Security*, New York, 2011.
- [51] T. Park, “Cosmo Bootswatch,” [Online]. Available: <https://bootswatch.com/cosmo/>. [Accessed 10 September 2015].
- [52] Microsoft, “Modern design at Microsoft,” [Online]. Available: <https://www.microsoft.com/en-us/stories/design/>. [Accessed 16 January 2016].
- [53] RFTrail, “Better Data Center It asset Management through smart RFID-enabled software,” November 2012. [Online]. Available: http://ws.iaitam.org/Misc/RFTrail_AssetTracking_WP.pdf. [Accessed 22 November 2015].
- [54] X. Yu-ning, “Research on NFC and SIMpass Based Application,” in *International Conference on Management and Service Science (MASS)*, Wuhan, 2009.
- [55] Blancco, “Protecting Your Enterprise with an Effective Data Erasure Strategy: Steps to Ensure the Removal of Sensitive Information When IT Asset Ownership Changes,” July 2009. [Online]. Available: http://ws.iaitam.org/Misc/Blancco_Protecting_WP.pdf. [Accessed 22 November 2016].
- [56] A. Ronacher, “Flask Extension Development,” [Online]. Available: <http://flask.pocoo.org/docs/0.10/extensiondev/>. [Accessed 20 May 2016].
- [57] A. Ronacher, “Flask Extensions,” 20 May 2016. [Online]. Available: <http://flask.pocoo.org/extensions/>.
- [58] eReuse.org, “devicehub-doc,” 1 May 2016. [Online]. Available: <https://github.com/eReuse/devicehub-doc>. [Accessed 1 May 2016].
- [59] M. Schuelp, C. Hagelueken, R. Kuehr and e. al., “Recycling from e-waste to resources,” United Nations Environ Program United Nations Uni, 2009.
- [60] C. Périou, “Waste: the challenges facing developing countries,” *Proparco*, 10 2012.
- [61] Basel Convention, “Ghana e-Waste Country Assessment,” 2011.
- [62] Basel Convention, “Where are WEee in Africa? Findings from the Basel Convention E-waste Africa Programme,” 2011.

- [63] European Commission, “Flash Eurobarometer 388, Attitudes of Europeans towards Waste Management and Resource Efficiency, European Commission,” *Eurobarometer*, 2014.
- [64] C. Felber, *La economía del bien común*, Deusto S.A. Ediciones, 2012.
- [65] EnviroInfo & ICT4S, “Best paper winners,” 9 September 2015. [Online]. Available: <http://enviroinfo2015.org/presentations/best-paper-nominees/>. [Accessed 20 May 2016].
- [66] D. Franquesa, Ó. Fabian, X. Bustamante, L. Navarro, D. López and S. Lamora, “eReuse.org: an ecosystem for traceable reuse of digital devices in a circular economy,” 2015.
- [67] National Telecommunications and Information Administration, “FALLING THROUGH THE NET: DEFINING THE DIGITAL DIVIDE,” 1999.
- [68] J. Ferber, *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Addison Wesley, 1999.
- [69] F. Ricci, L. Rokach and B. Shapira, *Introduction to Recommender Systems Handbook*, Springer, 2011.
- [70] Ajuntament de Barcelona, “Informació general - La gestió dels residus dels Punts Verds,” [Online]. Available: http://www.sostenibilitatbcn.cat/Preguntes_frequents/index.php?option=com_content&view=category&id=6&layout=blog&Itemid=23#3.1.. [Accessed 10 2015].
- [71] The european parliament and the council of the european union, “Directive 2012/19/eu of the european parliament and of the council of 4 july 2012 on waste electrical and electronic equipment (weee),” 4 July 2012. [Online]. Available: <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32012L0019>. [Accessed 10 2015].
- [72] Oxford University Press, “Definition of Recycle,” [Online]. Available: <http://www.oxforddictionaries.com/definition/english/recycle>. [Accessed March 2016].
- [73] P. Glavic and R. Lukman, “Review of sustainability terms and their definitions,” *Journal of Cleaner*, p. 11, 3 February 2007.
- [74] R. Studer, V. R. Benjamins and D. Fensel, “Knowledge Engineering: Principles and Methods,” *Data & Knowledge Engineering*, p. 25, 1998.

- [75] Schema.org, “Schema.org,” [Online]. Available: <https://schema.org/docs/about.html>.
- [76] M. Hepp and M. Sopek. [Online]. Available: <https://www.w3.org/community/gao/>.
- [77] F. Sánchez Carracedo, López and David, “El programa UPC-ReuTilitza: reutilización de ordenadores como metodología de aprendizaje-servicio para incorporar sostenibilidad, cooperación y economía circular en estudios TIC,” in *(Working paper)*, 2016.
- [78] W3C, “Editing the Web: Detecting the Lost Update Problem Using Unreserved Checkout,” 10 May 1999. [Online]. Available: <http://www.w3.org/1999/04/Editing/>. [Accessed 20 May 2016].
- [79] European Environmental Agency, “Glossary,” [Online]. Available: http://glossary.eea.europa.eu/terminology/concept_html?term=reuse. [Accessed 20 May 2016].
- [80] S. Fazle Rahman, “Modals,” in *Jump Start Bootstrap*, Collingwood, SitePoint Pty. Ltd., 2014, p. 133.
- [81] JEDI, “JEDI,” [Online]. Available: <http://jediupc.com>.

10 ANNEX

Please, see the *additional material* included with this project. There are the following files and folders:

- The in-code documentation and the specification of the API endpoints of DeviceHub. To access it, open the *DeviceHub documentation* folder, and then open the file *index.html* with a web browser.
- Results of the testing of serial numbers. Open the *Serial numbers tests.xlsx* with Microsoft Excel or compatible.