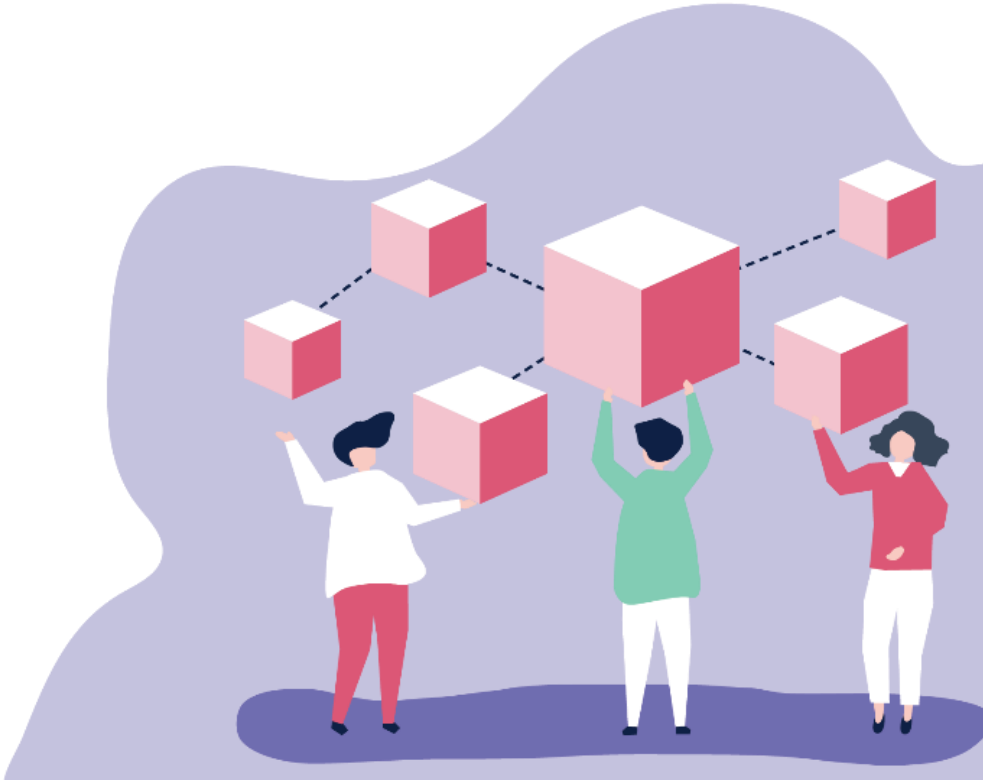
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 857223

GATEKEEPER

D5.1 Programming Interfaces for Dynamic Services Integration

Deliverable No.	D5.1	**Due Date**	31/03/2021
Description	Develop the programming interfaces for integration of the technologies and software components developed in WP4 and WP5		
Type	Other	**Dissemination Level**	PU
Work Package No.	WP5	**Work Package Title**	Integrated Plug&Play Gatekeeper Dynamic Intervention services
Version	1.0	**Status**	Final



Authors

Name and surname	Partner name	e-mail
Alba Gallego	UPM	agallego@lst.tfo.upm.es
Eugenio Gaeta	UPM	eugenio.gaeta@lst.tfo.upm.es
Alvaro Belmar	UPM	abelmar@lst.tfo.upm.es
Eduardo Buhid	UPM	ebuhid@lst.tfo.upm.es
Susanna Laurin	FUNKA	susanna.laurin@funka.com
Melad Munther	FUNKA	melad.munther@funka.com

History

Date	Version	Change
01/02/2021	0.1	Initial draft
01/03/2021	0.5	Contribution from partners
24/03/2021	0.9	Version ready for internal review
09/04/2021	1.0	Final version

Key data

Keywords	API, Developer Portal, WoT-TD
Lead Editor	UPM
Internal Reviewer(s)	Cristiano Pagetti (OK), Dave Raggett (W3C)

Abstract

This deliverable reports on the progress of T5.1 with the main goal of describing the design and development of the GATEKEEPER Developer Portal that will establish the way of

integrating APIs as digital things. A digital thing is intended as a digital twin representation of a device, service or data set that can be accessed through APIs.

As initial step, the working methodology is described, in addition it will be defined the mock-ups of the portal that provides the expected functionalities for developer users. Finally, development strategy and the portal code will be shared.

This deliverable is a live document that will be updated in a second version by M24.

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of contents

Abbreviations

Table 1: List of abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
FHIR	Fast Healthcare Interoperability Resources
GTA	Gatekeeper Trust Authority
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
MVVM	Model View View-Model
RDF	Resource Description Framework
SPA	Single Page App
TD	Thing Description
TMS	Thing Management System
UML	Unified Modelling Language
UX	User experience
W3C	World Wide Web Consortium
WoT	Web of Things

List of tables

List of figures

1 Introduction

This deliverable is describing the design and development of the Developer Portal associated to GATEKEEPER platform. The GATEKEEPER developer web portal will provide users registered as developers with the ability of building novel software solutions by using application programming interfaces (API), in order to dynamic integrate components and software developed in WP4 and WP5 within the GATEKEEPER platform.

GATEKEEPER platform components are described with Thing Descriptions (TD). A TD provides a digital twin representation of a service, a device, a dataset or a more general physical thing. This digital representation follows the W3C Web of Things standard[1].

Specifically, a TD describes:

- what the component is (linking also semantic meaning of the thing through JSON-LD standard);
- how a component could be accessed (through a set of standardized security definitions that describe the authorization and authentication flow in order to access the thing);
- how a component can be used (TD includes the definition of interaction patterns that are used for the description of the API associated to the thing).

Within the portal, a developer will have access to an easy-to-use web environment that provides a rich user interface designed to manage things, learn about the things and test the interaction with them.

The GATEKEEPER developer portal is mainly a front-end web application where developers can:

- see the things that they have created and published within the GATEKEEPER platform;
- see the things that they have access to use;
- see the things that they can buy;
- see the learning material about the things they are using;
- see create and manage applications where they can use things.

[1] Web of Things standard, https://www.w3.org/WoT/documentation/ , Last access March 2021

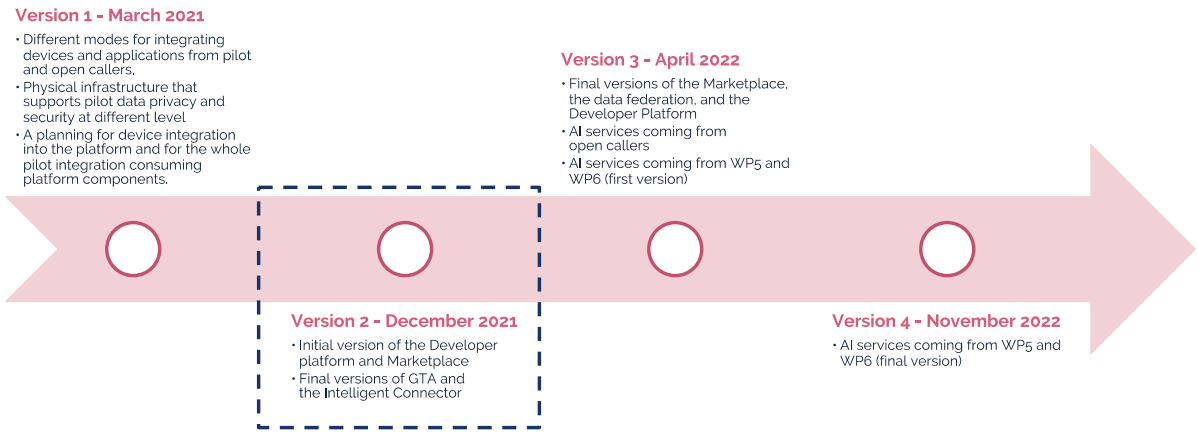


Figure 1. Gatekeeper platform roadmap

Following the Gatekeeper platform release roadmap in Figure 1, the Gatekeeper developer portal is expected to be delivered in the Gatekeeper platform version 2 expected by the end of 2021.

1.1 Structure of the report

This report includes 5 sections. Section 1 is an overview of the Gatekeeper Developer Platform functionalities that will allow services integration within the Gatekeeper platform. Section 2 describes the relation of the current deliverable and the other ones of the project.

Section 3 and 4 describe the methodology used for the design of the Developer Portal and the associated functionalities.

Section 5 describes the resulting mock-up of the portal as well as the initial development based on the prototype of section 4.

2 Relationship to other GATEKEEPER deliverables

GATEKEEPER Developer Portal is strongly integrated with the TMS, the GTA and the GATEKEEPER marketplace.

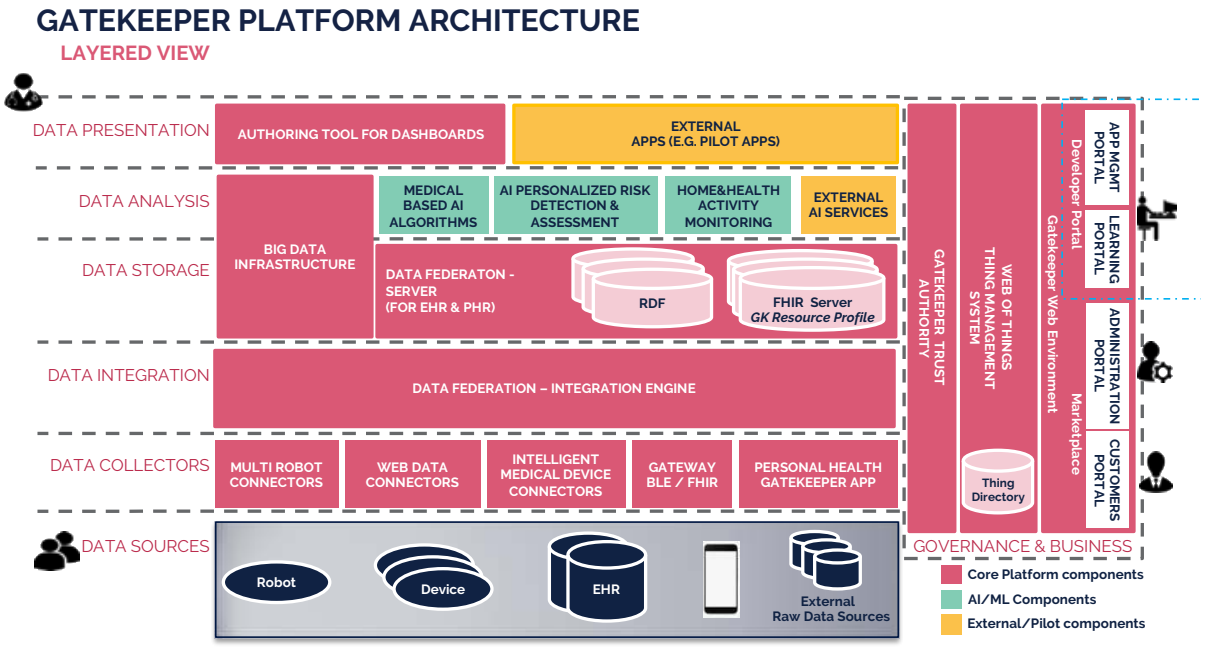


Figure 2. GATEKEEPER platform architecture

Figure 2 shows the logical view of the GATEKEEPER platform (described in the D3.2.2). Within this figure, the relations among the core components of the platform are shown. The Developer Portal is part of the GATEKEEPER web environment and share functionalities and look and feel with the marketplace (D4.6). Both GATEKEEPER developer portal and marketplace are client of the services provided by the TMS described in the deliverable D4.2 and the GTA described in the deliverable D4.5.

3 Methodology

Overall approach. A programming interface for the integration of services may seem like a user interface that would be highly technical by nature. But for the project results to be sustainable, the interface must be intuitive and easy to use for any stakeholder, no matter their previous experience or knowledge about the specifics of the project. Given that the GATEKEEPER platform will serve many different types of users, it has high ambitions when it comes to universal design in all parts.

In order to determine which features and functionalities users expect to find on the developer portal, a co-creation approach based on the survey method was selected. According to Business Research Methodology, the survey method [1] "*is used to test concepts, reflect the attitude of people, establish the level of customer satisfaction and conduct segmentation research*". In this case, an online survey was circulated to be filled by developers without any support.

In parallel, an analysis was made of existing developer platforms such as the Samsung developer platform, and the Facebook developer platform, to extract the main features that the GATEKEEPER developer portal should include. With these two inputs, the UPM team drafted the desired set of functionalities, which is described in Section 4. This grid of functionalities formed the baseline for on-going work on the user interface, which was performed in an iterative process involving the technical team of UPM as well as UX-design and accessibility experts from Funka.

UX-design, or User experience design, is a human-first way of designing an interface [2]. The terminology has changed over the years as the field has been known as usability, user friendliness, usefulness and so on, and there is an ongoing debate in the community about the perfect definition.

While the graphical design has to do with the visual appearance, aesthetics or "look-and-feel", the UX-design seeks to support user behaviour with the interface. Therefore, it can be seen as more of a conceptual, or functional, design. It builds upon traditional human–computer interaction (HCI) design and extends it by addressing all aspects of a product or service as perceived by users.

That is why the use case of developers uploading and handling software solutions has been in focus. By using an iterative approach, we have strived to add value to the user through every element. Agile development is based on short phases of work and frequent reassessments. The agile approach includes a cross-functional and collaborative process where teams with different experiences and competencies contribute to continual improvement. [3] We have repeatedly analysed how complex system interaction can be slimmed down and still retain its functionality. Through this approach, we aim to deliver a fluid experience to as many users as possible.

3.1 UX-design rationale and objectives

The UX-design work performed is based on the principles of universal design, a concept of designing all products and services to be aesthetic and usable to the greatest extent possible by everyone, regardless of their age, ability, or status in life [4] The concept of universal design is based on the idea that if you have "all" potential user requirements in mind from the start of a project, the result will be possible to use for a much broader audience – hopefully all. Instead of developing for the mainstream, with the risk of being forced to add specific accommodations for users with different needs, the universal

design principles make sure that even extreme use cases are taken into account. What is necessary for some, is beneficial for all.

The concept of universal design can be divided into seven basic principles, that are to be applied to all decisions made in the design process:

- Perceptible information
- Simple and intuitive use
- Approach (size and space)
- Low physical effort
- Tolerance for error
- Flexibility in use
- Equitable use

The first part of the UX-design process consisted of a general review of the mock-up and its different features. In this phase, we were examining broad elements based on the use cases described by the task leaders. Items considered were for example the user flow, how well the navigation supports the user through its interface and the accessibility of generic objects such as buttons and links. After remediating the overall framework to be more aligned with the user scenario, we turned to the details.

In this process, we looked at what type of information is displayed at any given moment and how the different parts of the content are balanced when it comes to size and placement. The focus was on making the interface and workflow as intuitive and straightforward as possible, without losing neither functionality nor flexibility.

An internal site map was constructed to understand the underlying logic of the workflow and the relationship between objects. This is a well-established method to ensure that items are placed and presented in a logical order, possible to find through more than one user behaviour and that all necessary steps and information are provided to the user in the expected order. For this purpose, we have added breadcrumbs to enhance navigation possibilities that fit a variety of user needs.

As an integrated part of the work done, the mock-up delivered, presented in section 5, provides the basis for EN301549/WCAG 2.1 compliance when developed. The EN301549 standard [5] serves as the minimum requirements for presumed conformance of the EU Web Accessibility Directive [6] covering all public sector websites. On the other hand, the Web Content Accessibility Guidelines (WCAG) are global guidelines for accessibility of web interfaces that the harmonised EN-standard points to [7].

4 Features of the GATEKEEPER developer portal

This chapter describes the result of the activities described in the previous section, in order to extract the main characteristics and use cases of the GATEKEEPER developer portal. To this end, the first step has consisted of identifying the stakeholders who will interact with the developer portal. In this case, only the GATEKEEPER developers will use this tool.

Once identified the potential users, it is necessary to know what type of functionalities these users expect to find. As described in D2.3, to extract all developer portal requirements and understand the user needs, online forms were circulated. These online forms that developers had to answer included some general questions about their expertise in semantic technologies, API platform or payments options, and on the other hand, an open question about how developers would like to use the GATEKEEPER developer portal. As a result, six functional and usability requirements were identified. Table 1, are presented these requirements with a description of how each requirement is mapped as functionality in the GATEKEEPER developer portal.

Table 1. User requirements extracted from surveys

Requirement	Rationale	Functionality in the GATEKEEPER developer portal
Account creation and management	Management the access to the platform.	On the login page, it will appear as a registration menu. On the home screen, it will appear as a profile button to manage user data
Login feature	Allows the user to access the customised functionalities according to his/her role	According to the user role, the view and features available will be different
Provision of an understandable documentation	Need to explain easily how to use the platform and Things, for example, using Swagger.	Each Thing will include a *getting started* section with useful information and first steps
Search Things available	Search Things by filtering (e.g., categories, names, …) to get more info.	On the home screen, a search component will appear. When the user wants to add things to an app, the user will be able to see Things filtered by categories.
In the case of the free account, there will be no requirement to provide a credit card	Free accounts should not require providing a credit card number.	It will not be necessary to include credit card
Provide developer portal as web-based	Needed to access easily the Things and their info.	The developer portal will be web-based and it will include a shortcut to access Things.

As reported in section 3.1, considering developers' reports in surveys and the outcomes of analysis of different platforms, it has been identified different use cases represented in

the UML diagram of Figure 3. This diagram graphically represents the interactions that the developer can perform when using the GATEKEEPER developer portal.

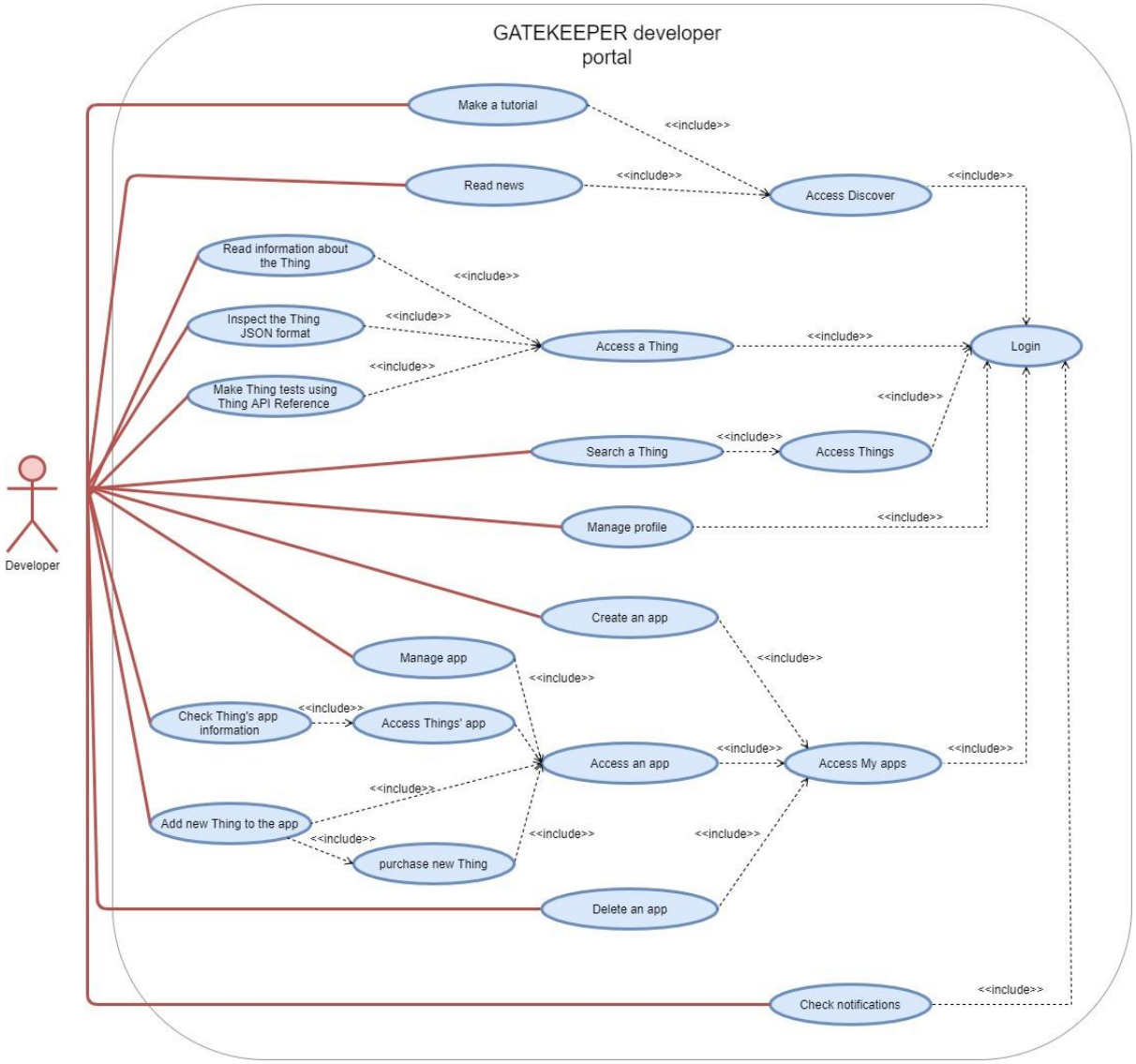


Figure 3. GATEKEEPER developer portal UML diagram

The use cases identified in Figure 3 are below described with the associated functionalities:

- Login/Logout: This functionality allows users to enter and leave the GATEKEEPER developer portal. According to the credential, the user will have available different kinds of functionalities. For the login, the GATEKEEPER Trust Authority (GTA) service is necessary.
- Restore the password: If the user forgets the password, he/she should be able to request a new password.
- Translation: Users should have the possibility to select between different languages (Spanish, Greek, English, etc), preventing language from being a barrier to use the portal.
- Shortcuts: On all screens, users should be able to go to the Home screen and main functionalities.

- Profile management: Users should be able to edit their account configuration (username, password, associated email, etc.).
- Notifications: They are necessary to enhance user satisfaction and engagement. Through notifications, users receive only relevant information related to their apps, Things updates, etc.
- Make a tutorial: Developers request to provide understanding information. For this purpose, it is needed to include tutorials for learning how to use the developer portal or Things.
- Read the news: A news section can be included to maintain developers informed about different topics.
- Search/Filter a Thing: It is important to include the functionality of searching and filter by type (e.g., purchased/not purchased) of Things to facilitate users the access to a concrete Thing. To manage Things the TMS service is needed.
- Access to Things: Add a functionality that allows users to have a complete overview of a concrete Thing is indispensable. For example, the user should be able to access general information about the Thing, know how to start to manage this Thing, know its JSON format and perform some test with the API Reference. To manage Things the TMS service is needed.
- Create an app: When an app is created, it is necessary to specify its characteristics. An app is composed of Things, so the functionality of adding Things to an app should be included. On the other hand, Things should be purchased before being used, therefore the functionality of purchasing Things has to be included too. To manage Things the TMS service and Marketplace connection are needed.
- Manage an app: The entire app or its characteristics should be able to be edited or removing.

The next section presents the prototype of the GATEKEEPER developer portal, which provides a global overview of how, and where, the features previously described are integrated into the different screens.

5 Results

The result of the task is a well thought through mock-up that includes all the key aspects of the programming interfaces for the integration of the technologies and software components. Each part of the interface and its functionality are presented below.

5.1 Mock-up prototype

The first screen that the user finds when enters in the GATEKEEPER Developer Portal is the Login (in Figure 4), a sign-in interface with input fields, links and buttons. In this screen the main functionalities are:

- Login/Logout: This functionality allows users to enter the GATEKEEPER developer portal. According to the credential, the user will have available different kinds of functionalities. For the login, the GATEKEEPER Trust Authority (GTA) service is necessary.
- Restore the password: If the user forgets the password, he/she should be able to request a new password.
- Translation: Users should have the possibility to select between different languages (Spanish, Greek, English, etc), preventing language from being a barrier to use the portal.

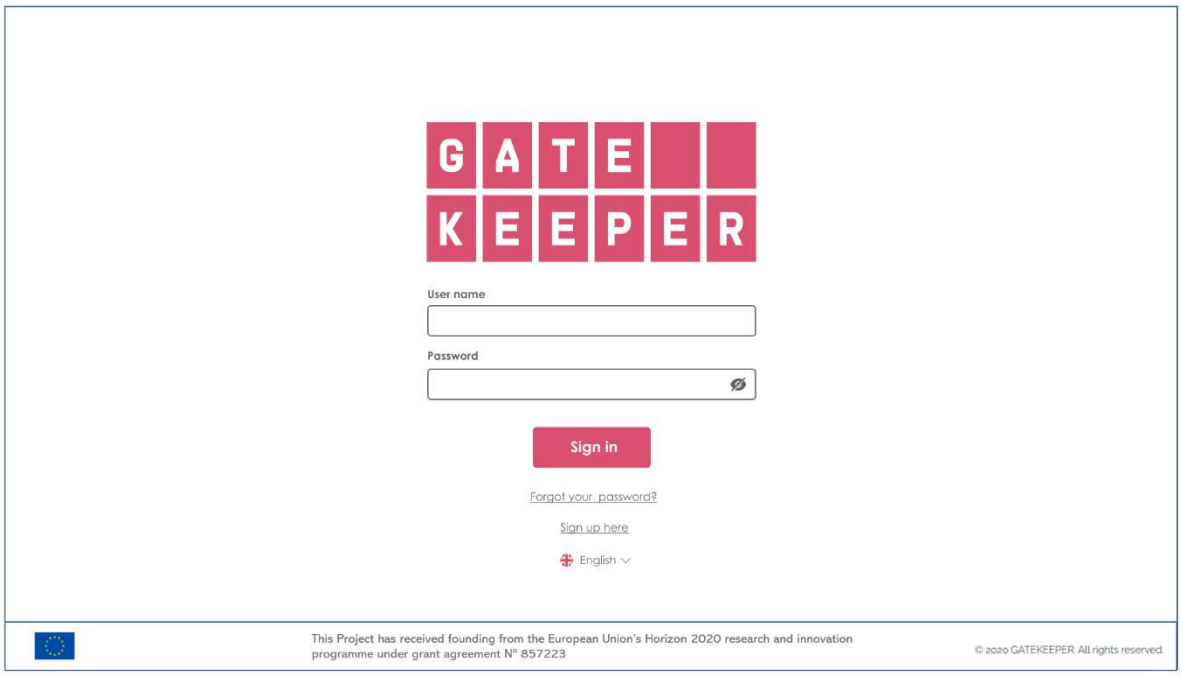


Figure 4. Login screen

When the user enters the credentials, the next screen is the Home screen (in Figure 5). This page consists of a header with a centralized search bar that is in line of sight of the user. Alongside are notifications and user profile page and below is the navigation bar followed by start page content. The user profile management button allows users to edit their account configuration (username, password, associated email, etc.). On the other hand, notifications (in Figure 6) are necessary to enhance user satisfaction and engagement. Through notifications, users receive only relevant information related to their apps, Things updates, etc.

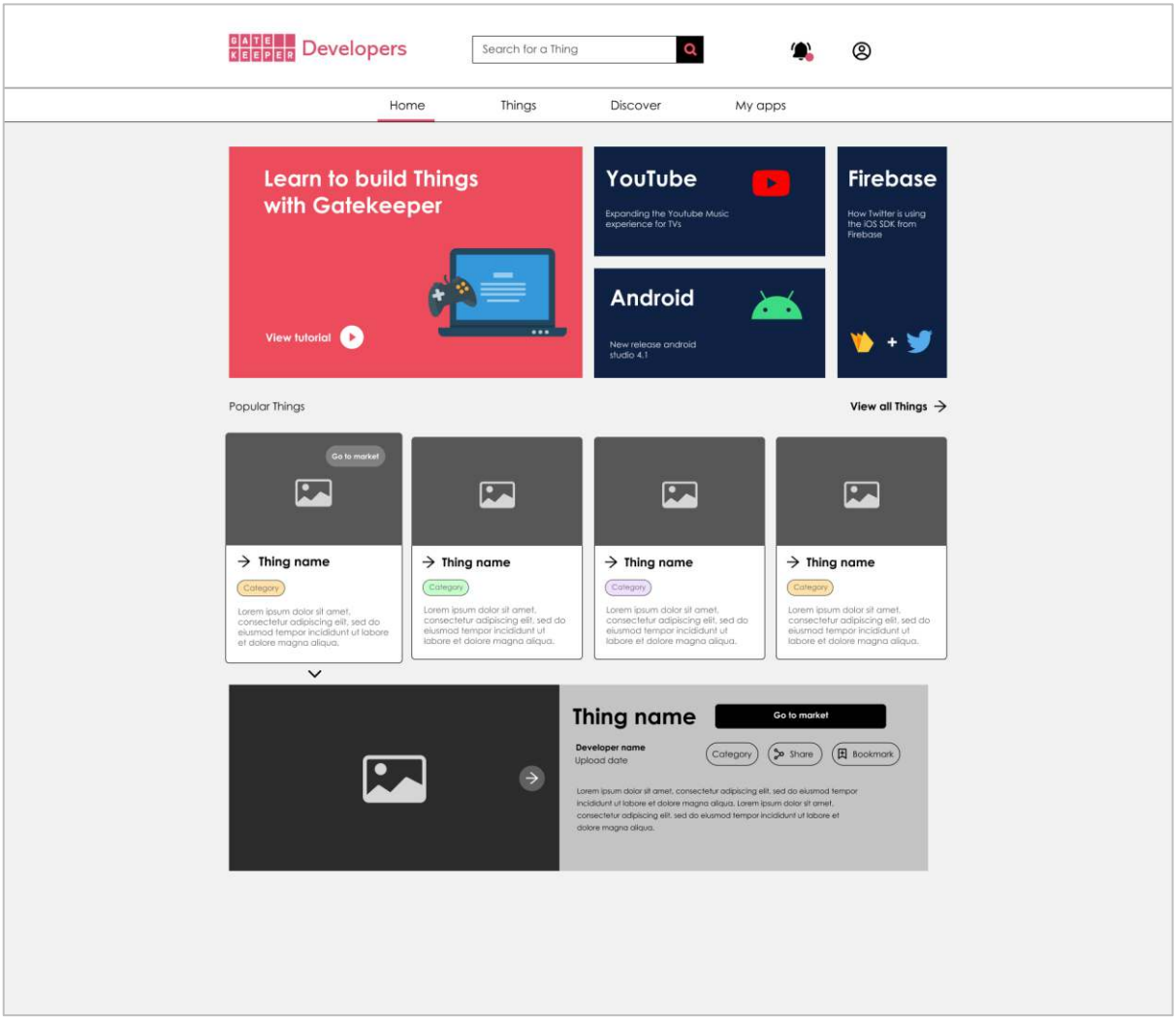


Figure 5. Home screen

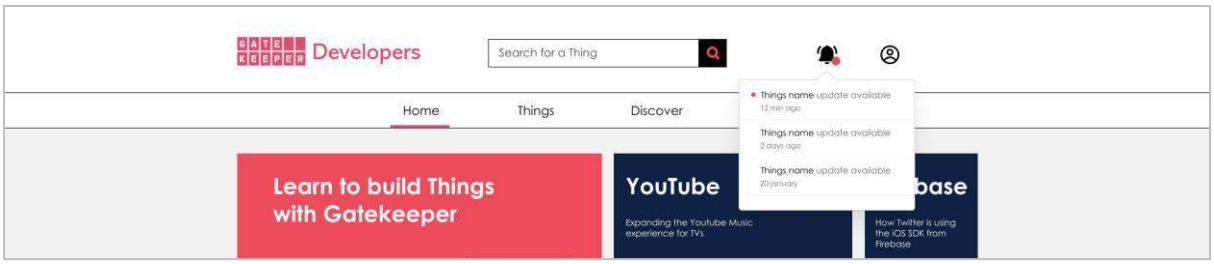


Figure 6. Home screen with notifications unfolded

The profile management, in Figure 7, is used to overview and update personal information and to keep track of your history through statistics.

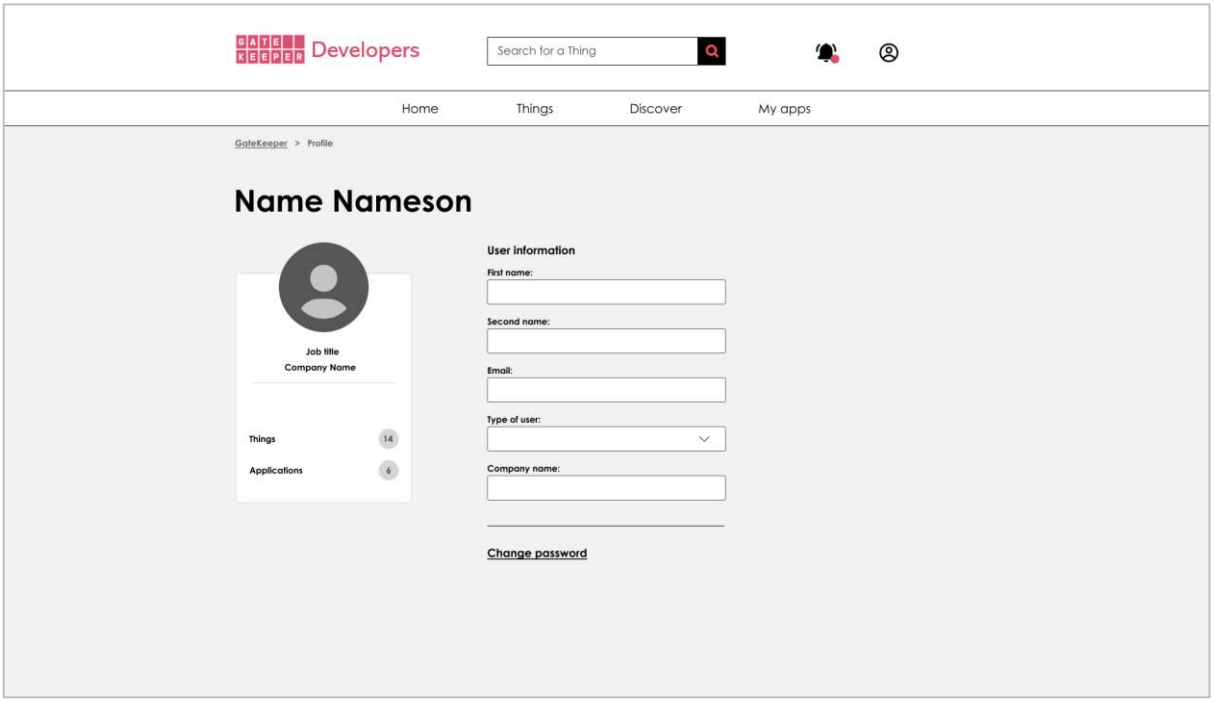


Figure 7. Home screen · Profile

Figure 8 presents the first block of the Home screen. These four cards promote to access the most recent news and other interesting content, such as tutorials of the use of the GATEKEEPER developer portal.

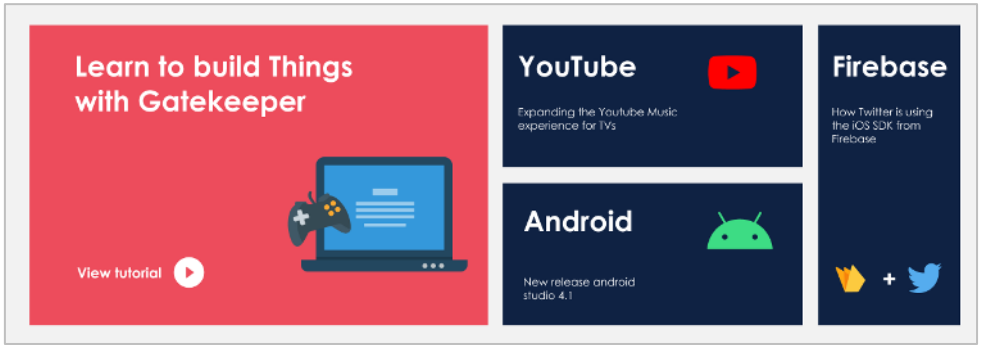


Figure 8. Home screen · Learn and News section

The home screen also includes quick access to the most popular Things (Figure 9), where each container represents a Thing. Each card contains an image, a title, the category of the Thing, and a short description. Upon hovering with the mouse, or when it gets keyboard focus, the card slightly expands and a shortcut to the marketplace appears in the top right corner to purchase the Thing. If the Thing has already been acquired, the shortcut will direct the user to the Thing page (in Figure 12).

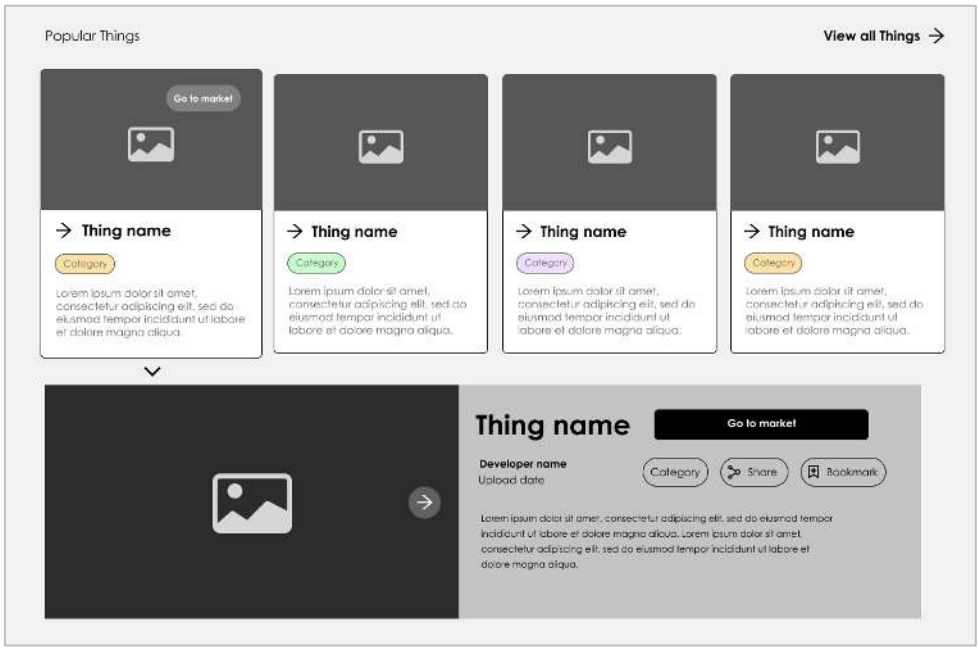


Figure 9. Home screen · Things management

When clicking on a card, a preview window folds below the card displaying more content, as Figure 10 shows. This lets the user view more screenshots, share and bookmark the Thing, and read a more thorough description before deciding to acquire the Thing.

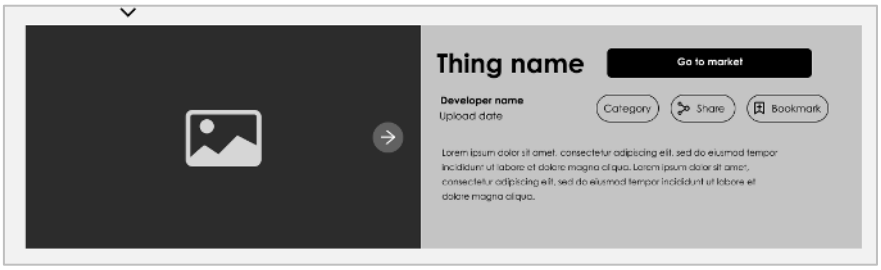


Figure 10. Home screen · Thing preview

The next shortcut is Things, in Figure 11, this screen contains information of all the Things available on the platform, sorted into different categories. Organized in the same familiar layout and grid, you can filter and sort the content to narrow down the options to your liking.

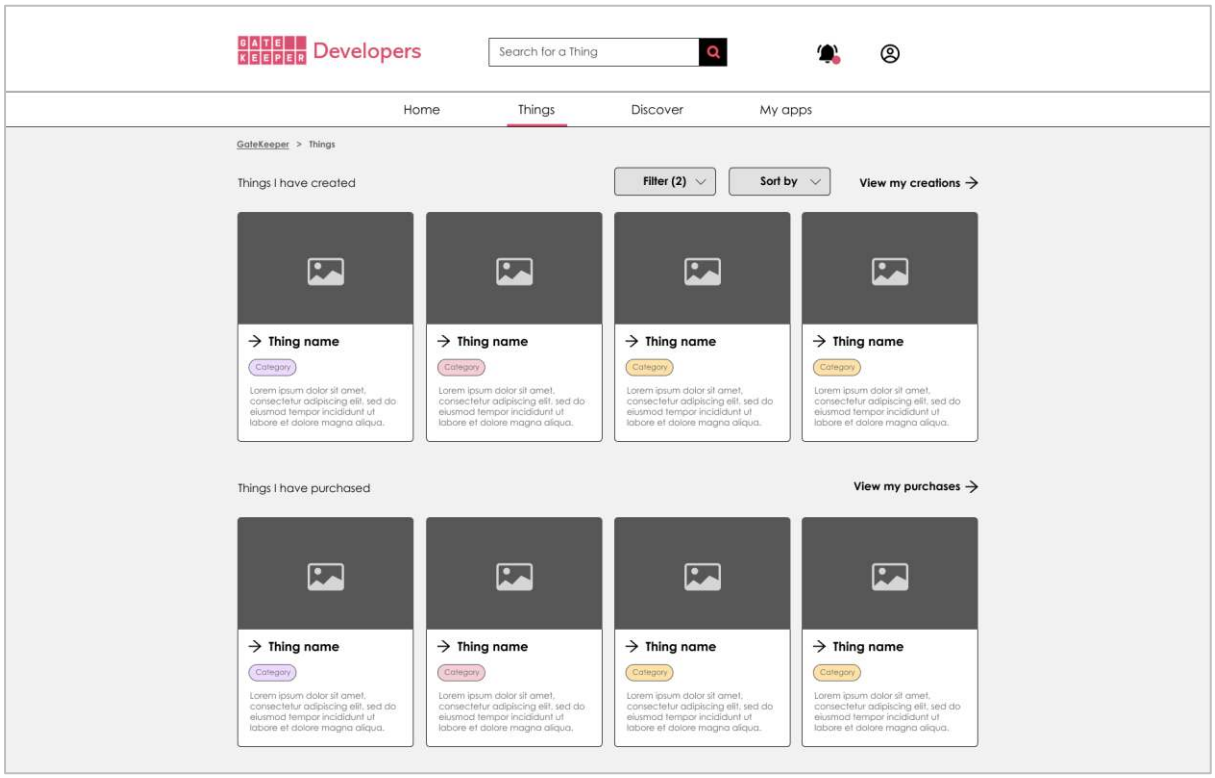


Figure 11. Things screen

When the user selects a purchased Thing, the next screen presented is the Thing overview (Figure 12). It is made by a stepper with 4 steps (Get started, Thing description, API and analytics). Through this screen, the user will be able to access general information about the Thing, know how to start to manage this Thing, know its JSON format, perform some test with the API Reference and access to analytics.
This section will be detailed in the next version of the deliverable providing integrated environment for testing the API that are associated to a thing description.

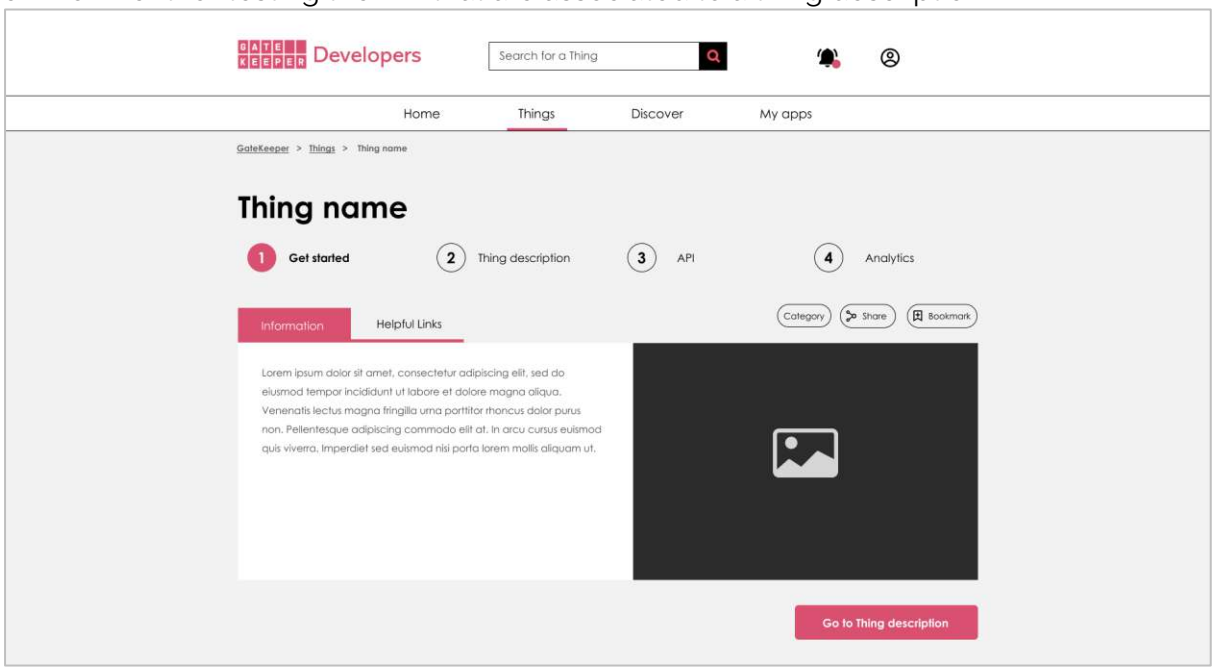


Figure 12. Things screen · Overview of the selected Thing (1)

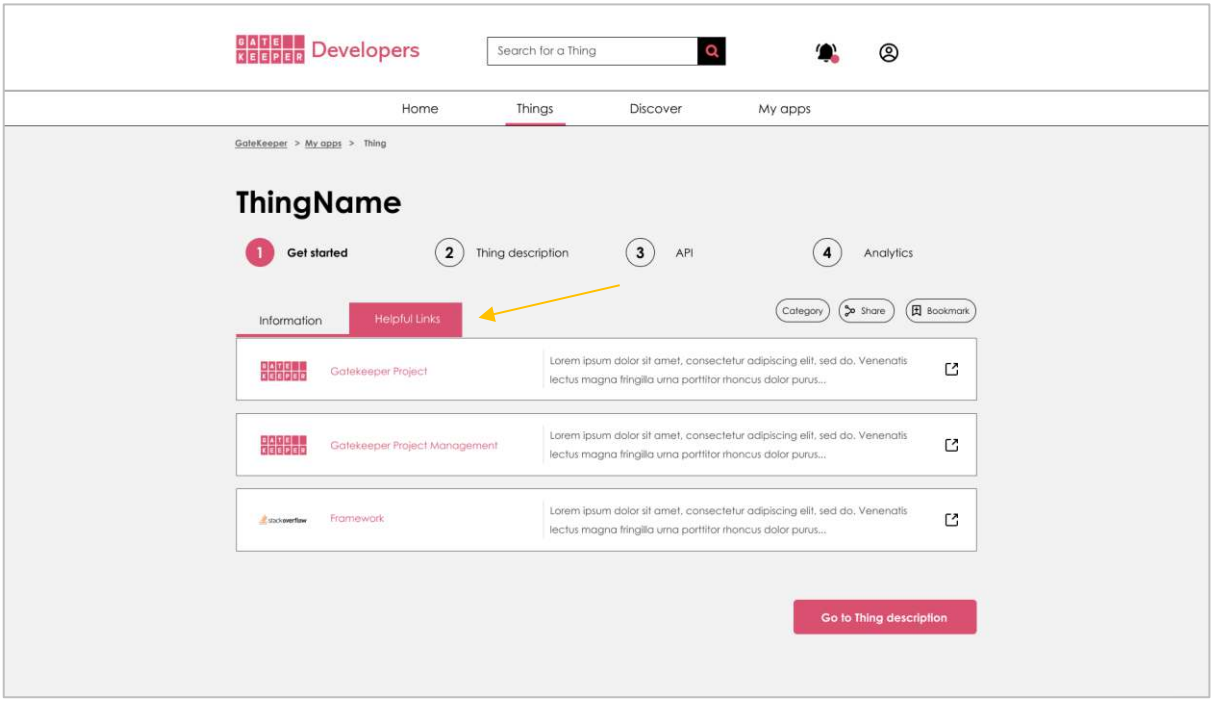


Figure 13. Things screen · Overview of the selected Thing (2)

Any related content that does not fit on this page can still be shared via “Helpful links”. Logos of the websites signal where each link leads to.

The third shortcut is Discover (Figure 14), a landing page for the tutorials and the news to give the user a brief overview. Here, users can catch up on the latest news and trending tutorials to keep up to speed. If you want to discover more, there are shortcuts to view more.

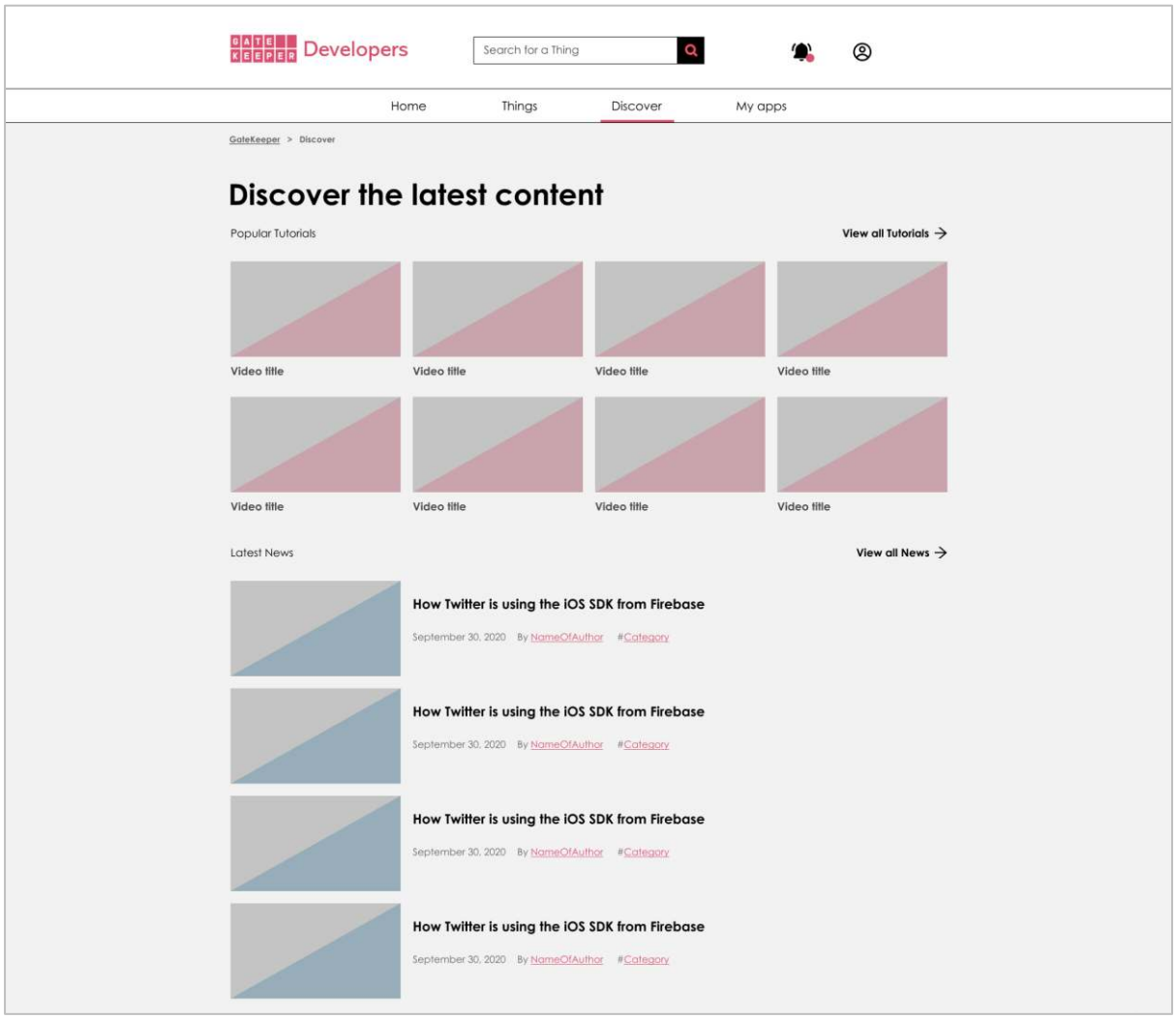


Figure 14. Discover screen

When selecting a news article, you will be directed to the article page to view the whole content, as Figure 15 shows. Each article has the publish date as well as the author's name and the category, which you also can click on to view more of.

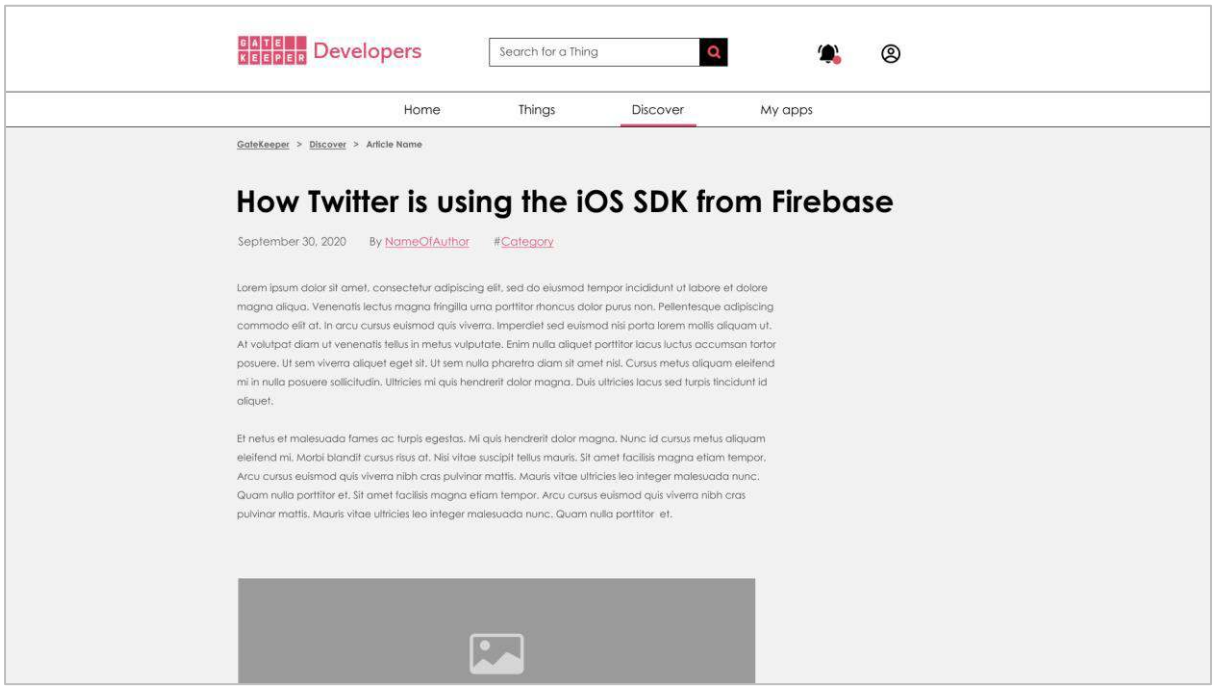


Figure 15. Discover screen · News

When selecting a video file, you will be directed to the video page which lets the user view the content selected as well as be inspired by other recommended videos (Figure 16).

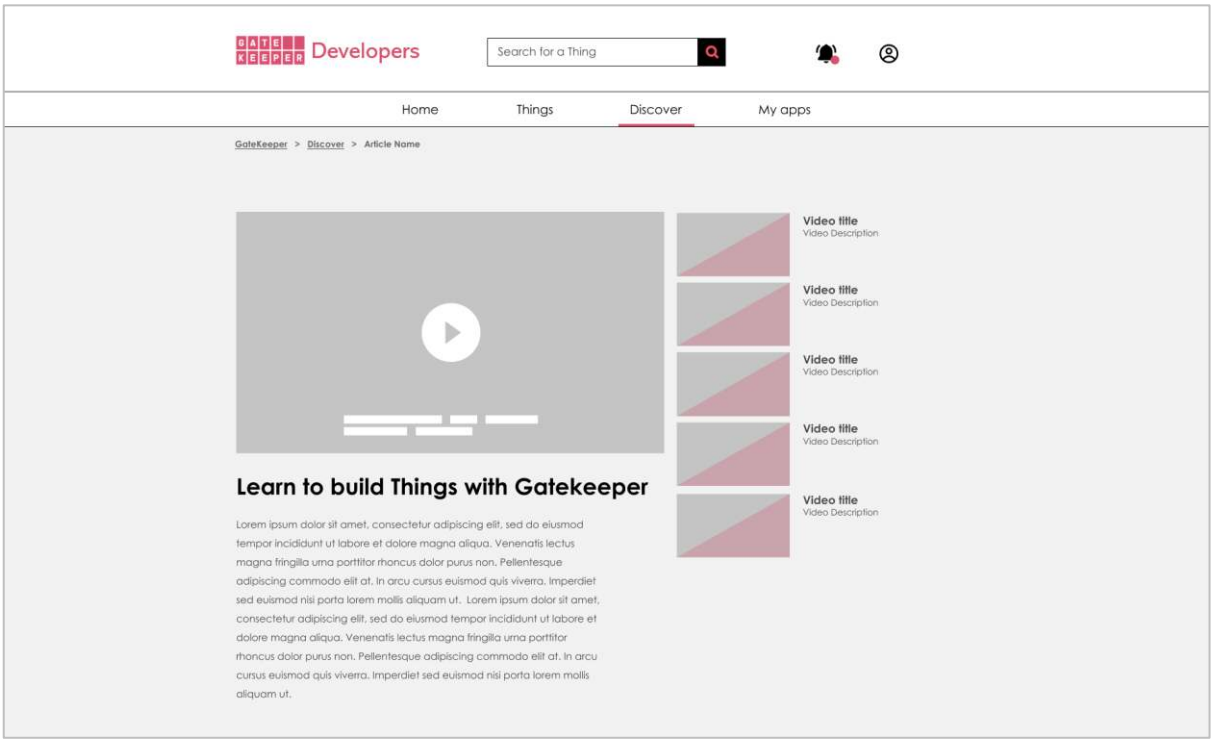


Figure 16. Discover screen · Tutorial

The last shortcut is My apps, in Figure 17, this page holds any application created by the user and allows him to create new ones. An application can contain different Things. A

preview of the number of Things inside an application is displayed on each card. Above the row of cards is a button to create a new application.

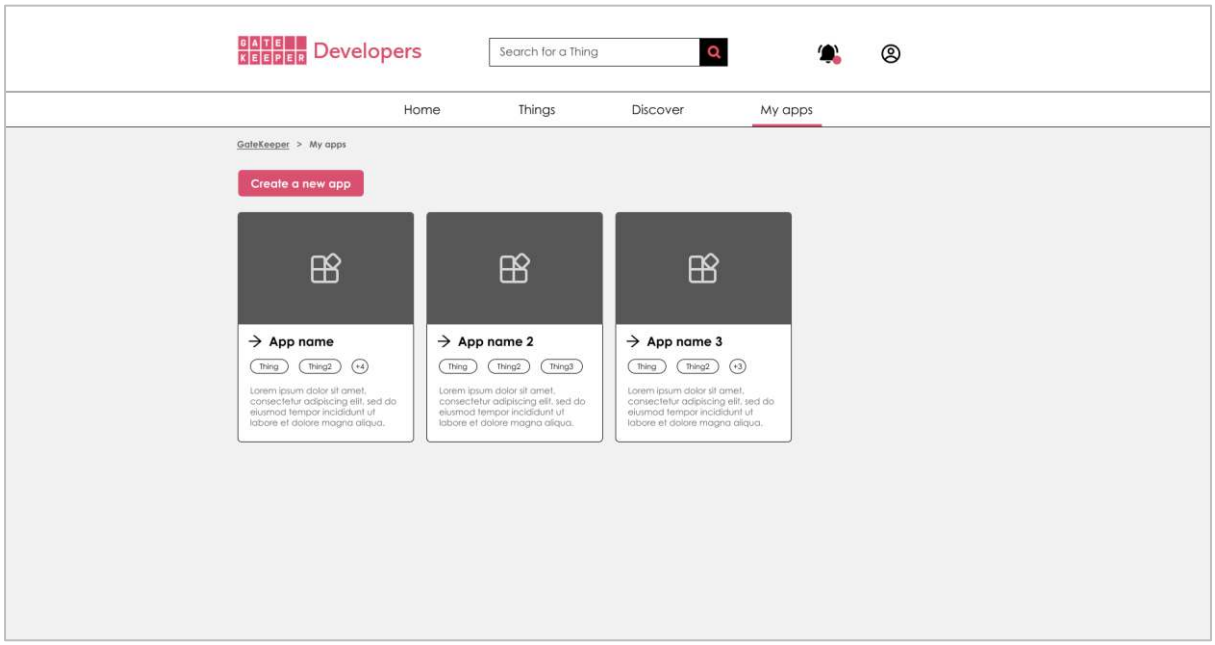


Figure 17. My apps screen

When clicking on one of the applications, the user gets a view of all the Things inside the application (Figure 18). You can also choose to add more things via the button "Add more things". This will direct to any acquired Thing and the possibility to purchase new ones at the marketplace. In the bottom right corner, the developer can manage the settings of the application and remove them.

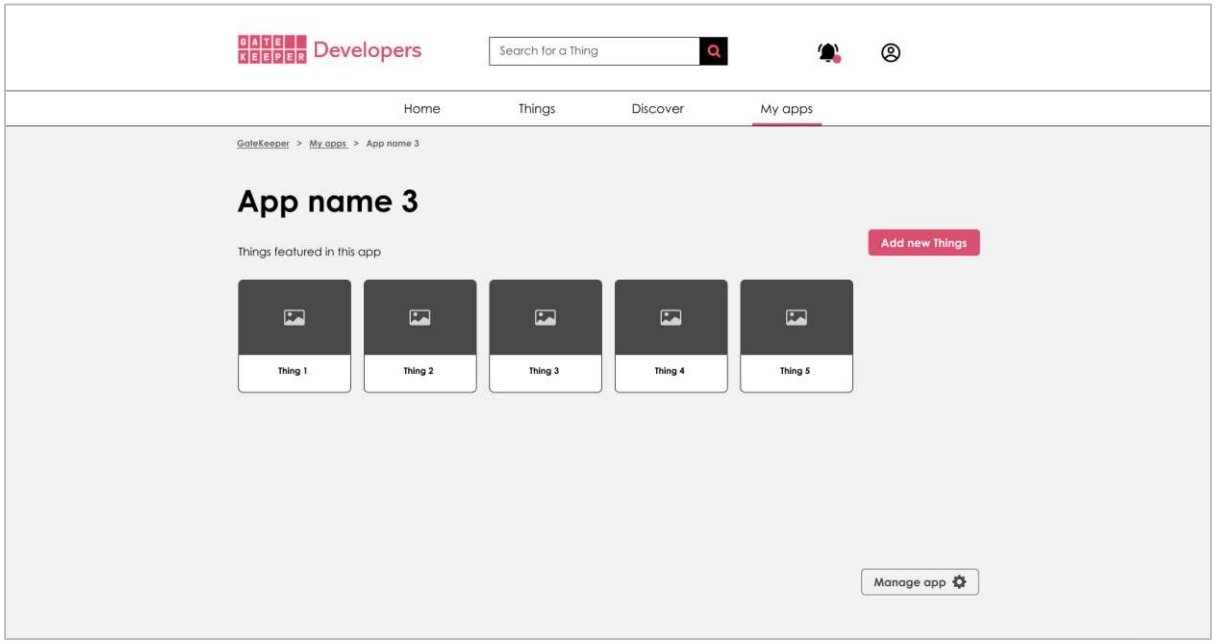


Figure 18. My apps screen · App selected

5.2 Development

For the development of the GATEKEEPER developer portal, we have selected the Angular framework [8]. Angular is a framework designed to build SPA (Single Page Applications) web applications using HTML and Typescript languages. The SPA strategy allows the application to be fully loaded at the first start, doing the application navigability quicker. Another advantage is that this framework is organized in modules, giving the possibility of adding new functionalities.

Angular implements the MVVM design architecture pattern (Model View View-Model), allowing two-way data binding between the view (HTML file) and the controller (typescript file), shown in Figure 19. In this pattern can be differentiated three components:

- Model: represents the data layer, holding the data information.
- View: responsible of display the data to the user.
- View-Model: intermediate point between model and view layers, taking care of the business logic.

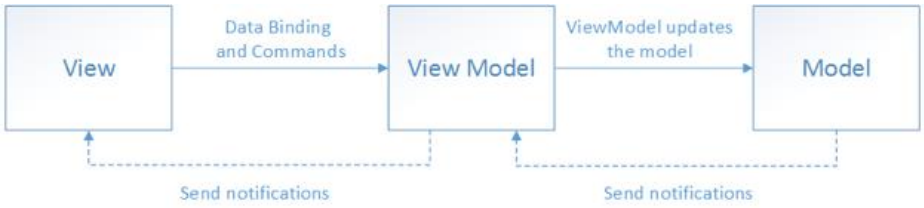


Figure 19. Design architecture pattern Model View View-Model [9]

As stated above, the Angular framework is organized in modules and, in turn, each module consists of a set of related components, directives, pipes and services, as Figure 20 shows. Each module includes the @NgModule decorator and is composed of the following blocks of functionality:

- Declarations: used to indicate the components which belong to the module.
- Imports: used to indicate the external needed modules to make the current module work.
- Exports: used to indicate exported components for the module, to be used inside other modules.
- Providers: indicate all the services to use inside the module.
- Bootstrap: indicate the main component to launch the application.

Any angular application has a principal module called AppModule. This module is in charge of giving the basic configuration to launch the entire application.

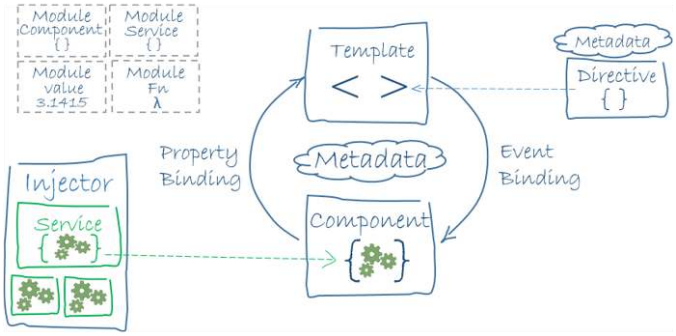


Figure 20. Angular architecture [8]

The main element used by Angular to build applications is called component, which is a set of HTML, TypeScript, and CSS files. The TypeScript file contains the business logic to control the HTML file, the HTML file renders the user interface in the browser, and the CSS file provides the style to the HTML file.

Any Angular application has at least one component called app-component, which is the root component of the application (normally added in bootstrap section in NgModule). Another relevant feature of this framework is that it allows adding plugins, such as Bootstrap or Material, providing the possibility of extending and adding more functionality to the application.

Once described the main characteristics of the framework selected to develop the GATEKEEPER developer portal, the next step is to define the Developer Portal component organization in Angular, shown in Figure 21.

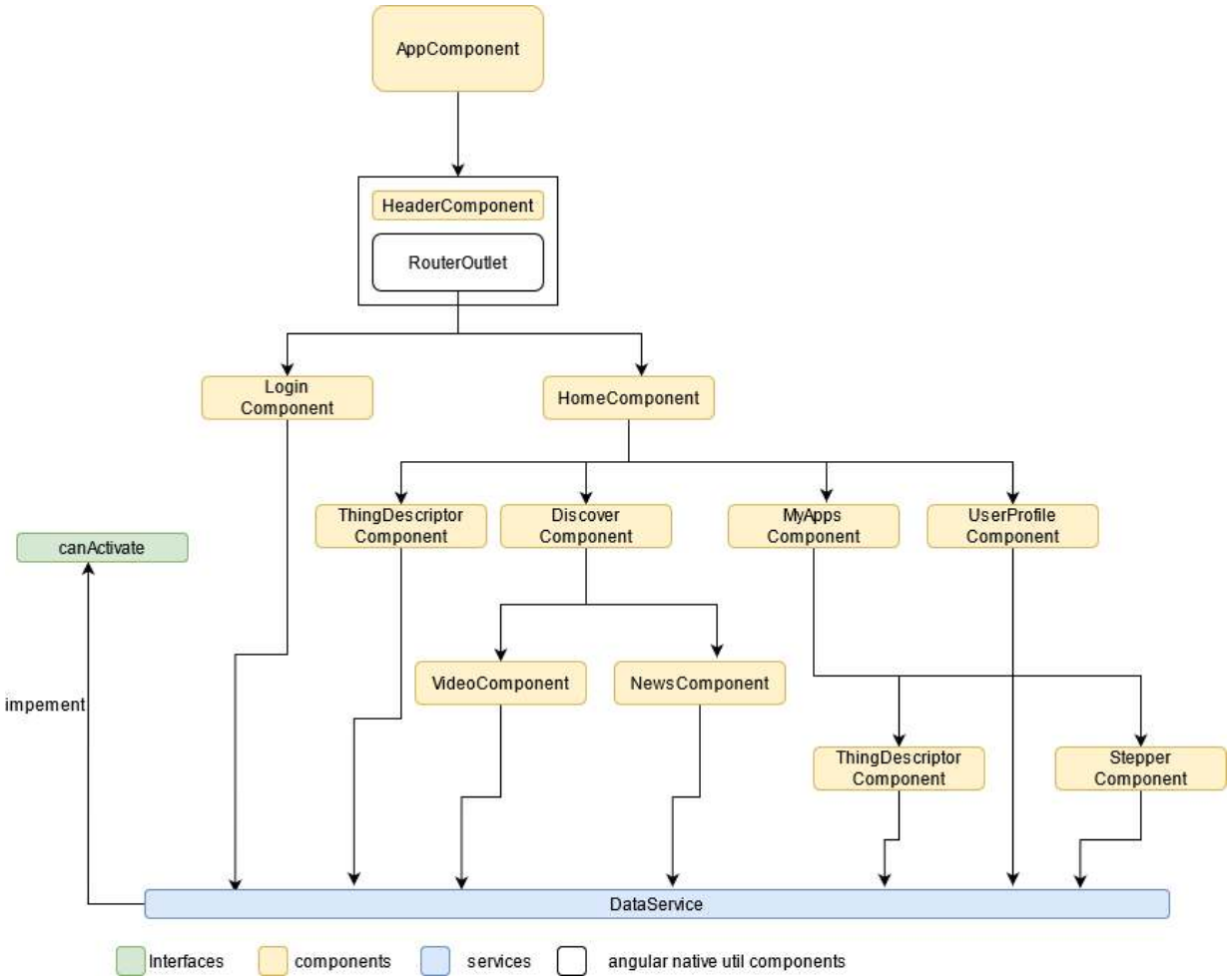


Figure 21. GATEKEEPER developer portal Angular architecture

In this Figure can be seen that each component is mapped with a functionality or screen of section 5.1.

- AppComponent: The main component loaded as root in the application. Inside this component is loaded the header component and the router-outlet.
- RouterOutlet: a placeholder that angular dynamically fills based on the current router status. This will be filed with LoginComponent or HomeComponent.
- HeaderComponent: Act as webpage header element.
- LoginComponent: Used for user authentication.
- HomeComponent: The main component is displayed as the homepage. It contains a set of other components to construct the design.
- ThingDescriptorComponent: A component to represent a ThingDescriptor. In the section things of the Home screen, the application displays a set of things.
- DiscoverComponent: Display the information in the discover section of the Home screen.
- MyAppsComponent: This component contains the functionalities to create, delete, edit and manage the applications owned by the user.
- UserProfileComponent: This shows the basic information about the logged user.

- VideoComponent: This component is in charge of playing a tutorial video inside the application.
- NewComponent: Used to display last news.
- StepperComponent: Manages the user guide step-by-step to read information about the thing descriptor and make tests with the API Reference.
- DataService: This is the service in charge of taking the data from the database through API calls. All components have access to this service to obtain the data about ThingDescriptors. This service also implements the canActivate interface to be used as RouterGuard, to secure the navigation in the application.

The source code of the GATEKEEPER developer portal is Open source and can be found in UPM Gitlab[2].

[2] Gatekeeper portal development project, https://gitlab.lst.tfo.upm.es/gatekeeper/cluster-demo/developerportal , Last access March 2021

6 Conclusions

This document describes the design and the implementation with the Angular framework of the initial version of the developer portal. The portal is designed on the requirements expressed by developer in the T2.3 and allow them to manage applications that include Things and its associated APIs.

In the view of each thing, it will be possible to test the interaction patterns associated with the thing compatible with Open API as well as explore the thing description in a tree-based view.

The final version of the developer platform is expected to be released in the version 2 of the Gatekeeper platform.

7 References

[1] S. L. Jackson, Research Methods and Statistics. A Critical Thinking Approach, Wadsworth, 2011.

[2] G. Allanwood and P. Beare, User Experience Design: Creating Designs Users Really Love,

London: Bloomsbury, 2014.

[3] T. Dingsøyr, S. Nerur, V. Balijepally and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software,* vol. 85, no. 6, pp. 1213-1221, 2012.

[4] M. Story, J. Mueller and R. Mace, The Universal Design File: Designing for People of All Ages and Abilities, Raleigh: North Carolina State University, 1998.

[5] ETSI, "EN 301 549 v2.1.2, Accessibility requirements for ICT products and services," 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf.

[6] EU, "Directive 2016/2102 on the accessibility of the websites and mobile applications of public sector bodies," [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016L2102&from=EN.

[7] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1," 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/.

[8] Angular, "Angular io," 2021. [Online]. Available: https://angular.io/guide/architecture. [Accessed 01 02 2021].

[9] Microsoft, "docs.microsoft," Microsoft, [Online]. Available: https://docs.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm-images/mvvm.png. [Accessed 01 02 2021].