CONFIGURATION TEMPLATE FOR INSTALLATIONS WITHIN WATTWIN PLATFORM

Chen Jieni Lin

Resumen– El proyecto consiste en la implementación de plantillas personalizadas en la plataforma de Wattwin. Este nuevo modelo permitirá que los usuarios puedan crearse instalaciones personalizadas, rompiendo con la limitación actual a instalaciones fotovoltaicas y de aerotermia. De este modo, los proyectos de los usuarios podrán adaptarse mejor a sus necesidades.
Para lograrlo, se deben modificar algunas entidades existentes en la aplicación y las relaciones entre ellas, lo que implicará cambios tanto en el front end como en el back end y la base de datos, así como en el motor de búsqueda. Es necesario comprobar que tras integrar la nueva funcionalidad, la plataforma siga funcionando correctamente.

Palabras clave – Fotovoltaica, Sistemas de energías renovables personalizados, MongoDB, Angular, Node

Abstract– The project consists on the incorporation of plant templates into the Wattwin platform. This new model will enable the application to provide customized installations for its customers, extending beyond the existing photovoltaic and aerothermal options. Hence, the projects more adaptable to their specific requirements.
To accomplish this objective, certain existing entities and their interrelationships need to be modified. These modifications will affect not only the front end but also the back end, the database, and the search engine. These changes are necessary to ensure the seamless operation of the entire platform, including the successful integration of the anticipated new functionality.

Keywords– Photovoltaic, personalized renewable energy systems, MongoDB, Angular, Node

1 INTRODUCTION

WATTWIN is a company that offers Software As A Service (SAAS), which is a Business Process Management (BPM) [1] platform focused on the design and commercialization of renewable energy installations.

The primary clientele of Wattwin comprises businesses engaged in the sale and installation of renewable resource systems to their own customers, such as EndesaX or Bonpreu Esclat. Wattwin's primary objective is to facilitate the entire process by offering a comprehensive range of non-integrated tools, starting from the initial stages of generating an offer to subsequent maintenance tasks. For example, the platform provides intuitive dashboards that enable users to monitor the progress of their ongoing projects. These dashboards offer a quick overview of the current status of each project, including steps such as client signature requirements, pending documentation, and completed tasks. Moreover, Wattwin allows users to personalize their projects by specifying the type of system (solar or aerothermal), the collaborating supply point company, and the proposed tariffs. Additionally, the platform can furnish information about the designated installation personnel along with their contact details, as well as an estimated budget based on the recommended tariff and actual consumption.

The company began by giving photovoltaic solutions and is continuing with aerothermal. However, to fulfill the constantly growing market demand, there is a need to widen nowadays' installations variety by including other services and products such as electric car charging stations or other energy management solutions.

• Contact mail: 1530245@uab.cat
• Specialization: Computer Engineering
• Tutored by: Miquel Àngel Senar (Department of Computer Architecture and Operating Systems.)
• Year: 2022/23

The primary objective of this project is to incorporate these services, which will be accomplished through the introduction of a new form of installation known as "personalized or custom installations." This approach allows users to create solution configurations that covers a wide range of projects, eliminating the existing limitations that are currently associated with photovoltaic and aerothermal systems.

1.1 Motivation

Wattwin fights against climate change. Being part of this revolution makes my work as a computer engineer worth more than just a few lines of code.

The existing workflow within our platform commences with the creation of a solution entity that covers various specifications, including the installation typology. Subsequently, they can generate customized budgets for potential installations, with each budget associated with a specific solution. Upon reaching this stage, the project can be created by assigning a solution to it. Consequently, within a single project, there exist multiple budgets, and multiple distinct installation. However, it is important to note that these installations must adhere to the same type, as they are constrained by the shared solution, which acts as a limiting factor regarding the typology.

With the increase in environmental awareness, the variety of products used to reach this aim is expanding too. To follow up with market demands, the company needs a new type of installation: personalized one. Nowadays, when a customer generates a process instance, entity which corresponds to a project or service, in the platform, it has a unique solution related to the typology: aerothermal or PV.

The aim of this project is to break this limitation as well as adding a new type of installations. It is sought that the installation does not inherit the solution typology but has its own one. This would mean that one process instance can have different kinds of installations. In the clientele perspective means that the user can offer its clients in one single project several type of renewable source systems, instead of creating one project for each type of solution.

To achieve the objective of decoupling the relationship between topology and installations, a new model known as a "plant template" is required. This class will serve, among others, the purpose of specifying the typology of the system as well as some compulsory products (this list is called bill of materials within this project), instead of relying on the solution. The budget and project components will still be associated with a solution. However, the key distinction lies in the fact that the solution will now have multiple distinct templates. As each installation is assigned a plant template, it means that a project can include various types of installations, depending on the allowed templates within the solution.

This modification will also facilitate the incorporation of a new category of installations, which are the desired "personalized" ones. If a template lacks any typology specifications, it means that it is a customized template. So that, the projects will be able to have either photovoltaic, aerothermal or custom installations if the solution is configured so that it allows all three types.

In summary, the proper execution of this project is expected to bring the platform a wider range of renewable installation types and it will offer a more generic type of project.

1.2 Objectives

The project is not independent of the rest of the platform; on the contrary, it requires integration with the existing components. Therefore, it is not only a matter of creating new modules but also modifying the ones already present and reassessing the relation between them.

The primary features to fulfill to reach the objective, ordered in descendant priority, are the following:

- Add a new type of installation type: personalized.
- Allow modify and configure the template.
- Clearly show the detail of the template.
- Maintain the consistency of existing modules in the database with the new ones.
- Adapt the existing modules to the template module.
- Adapt the components to the most recent design.

2 METHODOLOGY

Wattwin follows the Scrum [2] methodology, which is part of the Agile project management philosophy due to its iterative and incremental approach.

By following this methodology, the company is split into several teams of a small number of persons with a scrum master, who leads the team and organizes the meetings. In our situation, this person also plans the sprints by taking the tasks from the product backlog [1].

The scrum cycle has a duration of X weeks depending on what the company establishes, in my situation, it is of three weeks. Each cycle is formed by a scrum planning meeting, some daily scrum meetings, a sprint review, and a sprint retrospective meeting.

During the daily meetings, each person communicates the plan for the day, corrects what was done from the plan for the previous day, and explains the stoppers encountered if any.

In our situation, the sprint review and retrospective, as well as the sprint planning are joined together in one single meeting. So that, at the same time we end the sprint by telling the pro and cons of how the sprint has been carried out, we start the new one by evaluating the tasks by giving them story points[2] and assigning them to a developer. If there are tasks that have not been finished in one sprint, it usually persists in the following one.

The process of evaluation starts with explaining what the task is about. Once in context, each developer should individually weigh the task difficulty. In order not to influence each one's decision, everyone's opinion is kept in secrecy until everyone is ready, which is the moment when the final punctuation of the task is discussed. If all the members

[1] List of tasks for the development team that comes from the roadmap and its requirements.

[2] Unit of measure to express the amount of effort required to implement a task.

of the team rated the task with the same number of story points, then the mark is directly assigned. If not, the persons who have not agreed must discuss until reaching an agreement.

In addition, there are weekly follow-up meetings that take place per team, to show the progress of one's task to the rest of the team. This is done to have a global image of what is being accomplished.

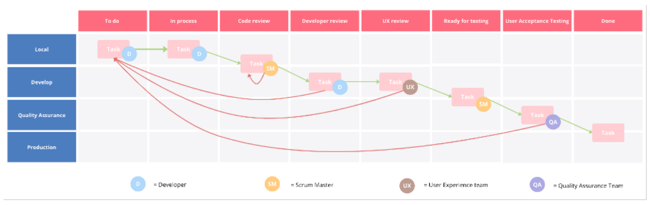


Fig. 1: Task pipeline

On the other hand, in reference to the task progress, the company follows a methodology in the form of the task pipeline as shown in Figure 1. Each task travels from different phases and servers before it is completely done.

All tasks that belong to the sprint start in the "To do" column until the developer begins the assignment in a local branch when it is moved to "In process". Once finished, the team member requests that one's task is merged into the platform source code; from this moment on, the tasks enter "Code Review" status. Meaning that it has to be checked by the scrum master. If no problem is found, then it is merged to Develop server and moved to "Developer Review"; otherwise, it relies upon the same stage until corrections are done.

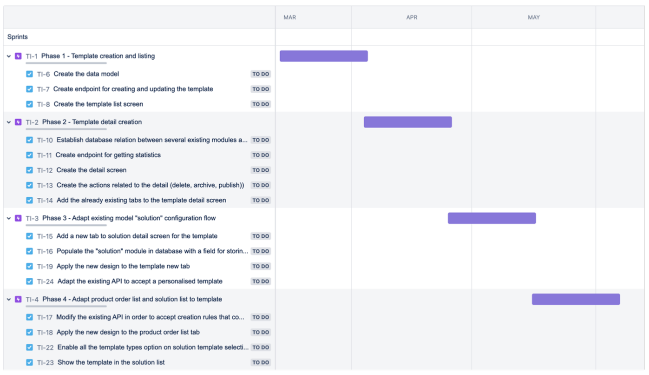
In the "developer review" phase of the pipeline, as its name indicates, the developer has to check whether the task has been completed accurately, even merged with other team members' work. Sometimes, at this moment the developer notices some missing requirements or functional compatibility with others' tasks; then, the task has to be restarted. When the programmer considers the task to be complete, it is moved to the next step, finishing the task if everything goes well.

The User Experience (UX) review phase takes place due to the need of testing the product, not only from a functional perspective but also from the user point of view, checking whether the interaction human-product is convenient. In Wattwin, there is a team more focused on this aspect, so this becomes their duty. If any error is encountered, the task is restarted; otherwise, it continues the flow of the pipeline arriving at "Ready for testing". This is a column where the task rests until a periodically-planned merge to Quality Assurance (QA) server takes place. This responsibility relies on the scrum master.

Last but not least, the task arrives at the "User Acceptance Testing (UAT)" phase; when the role in charge is the analyst of the quality assurance team, who guarantees the quality of the product. For instance, ensures that the platform follows a unified design pattern by checking that the developer has strictly followed the design model agreed at the design phase. This step is prior to the task backlog; thus, it does not appear in the task development pipeline. If any problem is encountered, the task goes back to the "To do" list; if not, it will wait in "Done" until the periodically-planned merge to the production server. The production server is the one that hosts the final customer requests; in other words, the final customer accesses this server.

The different servers coexist seeking progress in every next server. Local servers can be compared to a personal playground, where changes of any kind will not influence and have any repercussions. The development server, in Figure 1 named Develop, is like a shared playground, where personal errors are exposed to other developers working within this server. When it comes to the QA server, it should be more similar to the production server; prior errors should be cleaned.

3 Planning



Sprints

TI-1 Phase 1 - Template creation and listing
TI-6 Create the data model TO DO
TI-7 Create endpoint for creating and updating the template TO DO
TI-8 Create the template list screen TO DO
TI-2 Phase 2 - Template detail creation
TI-10 Establish database relation between several existing modules a... TO DO
TI-11 Create endpoint for getting statistics TO DO
TI-12 Create the detail screen TO DO
TI-13 Create the actions related to the detail (delete, archive, publish)) TO DO
TI-14 Add the already existing tabs to the template detail screen TO DO
TI-3 Phase 3 - Adapt existing model "solution" configuration flow
TI-15 Add a new tab to solution detail screen for the template TO DO
TI-16 Populate the "solution" module in database with a field for storin... TO DO
TI-19 Apply the new design to the template new tab TO DO
TI-24 Adapt the existing API to accept a personalised template TO DO
TI-4 Phase 4 - Adapt product order list and solution list to template
TI-17 Modify the existing API in order to accept creation rules that co... TO DO
TI-18 Apply the new design to the product order list tab TO DO
TI-22 Enable all the template types option on solution template selecti... TO DO
TI-23 Show the template in the solution list TO DO

Fig. 2: Gantt chart

The project is thought to be completed within three months. Since we work with scrum methodology, as previously explained in the last section, all the tasks will be divided into four sprints with a global distribution of tasks as illustrated in Figure 2, which is a Gantt chart [3]. This diagram is used as a tool to control the work completed compared to the planned. Each phase corresponds to a sprint mentioned, with its expected duration.

Nevertheless, since the number of story points is an agreement between all the members of the scrum team, it cannot be known beforehand the number of tasks that each scrum will have. Thus, the planning at the starting point of the project it is just personal.

4 Technologies

The project's front end[3] will be developed in Angular[4], which is a framework of TypeScript[5] that in turn, is a superset of JavaScript and adds optional static typing. JavaScript is a scripting language understood by all major web browsers.

When it comes to the back end[4], LoopBack[6] is used. This is an extensible Node.js[7] and TypeScript framework. Node.js is an asynchronous event-driven JavaScript runtime.

[3]Part of the website that the user interacts, visualizes, and experiences, such as the Graphical User Interface (GUI).

[4]Also called server side, it is the non-visual part of the website in charge of data management and storage. It enables APIs so that the front end can request processed data from the back end.

A framework is a structure that provides a set of extra tools, libraries, and conventions over a programming language. It usually facilitates the developer to organize the code or makes it easier and faster to write and understand.

MongoDB is the database to which our LoopBack APIs make their requests. It is classified as a NoSQL[8] or non-relational database, which means that it is not structured like relational ones: with table relations. It is simpler to design and scale horizontally.

However, databases are not always enough. To speed up some important searches, the platform uses ElasticSearch (further on named ES) search engine. This is a system that, if you provide the conditions that the data should meet, lists the matching information from the computer system.

To perform version control and project storage, the company makes use of Bitbucket, which is a Git-based source code repository.

5 INITIAL STATE OF THE PROJECT AND STATE OF THE ART

The platform can be seen as a system with a set of input data; after dealing with this data, it is able to generate an output, which is usually some kind of document (technical, offers, quotations, and the materials needed ...).

Nowadays, the input data is primarily focused on configurating a PV or Aerothermal project. For each of them, a specific installation design is needed. The first one requires a graphical design: a map is shown, the user searches the desired location and draws the surface of the solar panel and configures some parameters. While aerothermal one requires a preset form with some questions.

Having the initial design settings completed, the customer has available the initial bill of materials required for the installation which is inherited from the solution, a set of billing comparisons (showing the incomes, and savings…), a tariff configuration option …

Nevertheless, the platform has also other functionalities not directly related to our project, such as its own mail system, product catalog management, technician report and management, customisable customer offer pipelines (to control the state of all the projects that the customer is working with) …

When it comes to the state of the art of technologies used, Angular is at version 15. With the upgrades made, it has been improving its performance, its support for accessibility… It has assured compatibility with the latest versions of TypeScript while dropping support for the oldest ones. Some of the big companies that use Angular as one of the programming languages used include Microsoft (on its office suite product Office, for example) or Google (on its mail system Gmail).

The most recent version of LoopBack is number 4; with which this language has reached a catch-up with the latest technologies, it has removed complexity and some inconsistency that existed among modules, and it has introduced controllers and repositories for better composability.

6 REQUIREMENTS

This section will show the functional and non-functional requirements withdrawn in the project, split into back end related ones and front end related ones.

6.1 Functional requirements

The requirements which define a functionality within the software are called functional requirements. Some of the principal requirements of our project are the following:

TABLE 1: FUNCTIONAL REQUIREMENTS

1- Allow creating new templates	Back end
2- Allow updating the already existing templates	Back end
3- Allow obtaining statistics on the usage of the templates	Back end
4- Adapt the actual solution configuration flow to the templates	Back end
5- Allow obtaining the product order list rules configuration from the solution template	Back end
6- Allow obtaining the product tree from the solution template	Back end
7- When saving an installation with a template applied, it should store the default values set by the template.	Back end
8- Populate the actual solution data set with a plant template	Back end
9- Allow the visualization on a screen with a list of existing templates while giving the option to create a new one	Front end
10- Allow archiving and deleting existing template	Front end
11- When configuring the template, allow setting the option of enabling the energy supply point view in the solution that applies the corresponding template.	Front end
12- When configuring the template, allow setting the option of enabling technician view in the solution that applies the corresponding template.	Front end
13- When creating a new solution, enable to choose all the existing template types.	Front end
14- Show the template selected in the solution detail screen.	Front end

6.2 Non-functional requirements

Non-functional requirements are specifications that improve the main functionality of the system. Some of the ones extracted from our project are:

TABLE 2: NON-FUNCTIONAL REQUIREMENTS

1- Apply the unified styles of the platform to the new components	Front end
2- Separate the template-related functionalities in different screens organized by tabs	Front end
3- Give the option to show the templates list in a table mode	Front end
4- Responsive design of the screens	Front end

7 SYSTEM DESIGN

7.1 System classes

This subsection will illustrate the relationship between the existing classes compared to the newly created ones to fulfill the requirements.

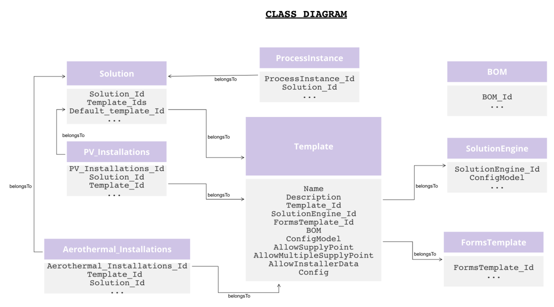


Fig. 3: New class diagram

The Figure 3 exposes the relation between the different existing classes in the system and how are they related with the new model created: Template.

Each class requires an identifier, which in the illustration is simplified as "Id", so that each instance can be singled out. The attributes that are not relevant to our project are not shown.

Our principal model has a unique name, an optional description, an identifier, some configuration data and three booleans[5] that represent what their name clearly indicates. The first flag points out whether supply points are allowed, and only in case they are, the second flag makes sense: whether is possible to set several of them. When it comes to the latest Boolean variable, it just shows if data related to the installer will be shown.

As it can be seen, the new class have two types of relations with other classes: belonging one and embedded one. The main difference between relies in where the data is stored. Template model stores the identifier of "solutionEngine" and "FormsTemplate"; however, these objects are stored in their own model. In short, it mentions an instance of another model.

On the other hand, Template class embeds an "Bill Of Materials (BOM)" object. This means that the instance of the object is stored directly in the Template class. So that, if one queries in the BOM document in the database, the instance stored in Template will not be found.

[5]Data type that can have only two possible values: "true" or "false".

It can be seen from the class diagram that each of the installation types has a Template instance's identifier. But when it comes to the solution model, it uses and therefore stores several Template identifiers, but it references one Template instance as default one. Obviously, this default instance must be included in the array of Templates stored.

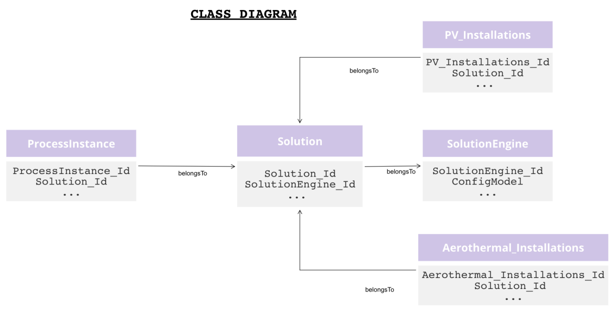


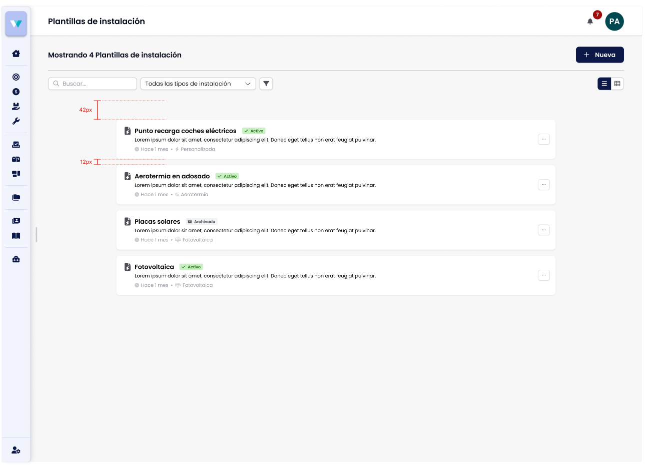
Fig. 4: Old class diagram

Figure 4 shows the part of nowadays system's class diagram that is important to our project. By comparing Figures 4 and 5, it can be clearly seen how the previously mentioned changes are put into practice.

The configuration model (configModel) attribute of the solution engine class is the variable that indicates the installation typology. Since the relation between solution and solution engine has dissolved and a solution accepts several template identifiers, it means that several types of solution engines are allowed per solution. That is why, despite maintaining the relationship between the solution and the installations, a process instance can still have different types of installations.

7.2 UX/UI design

In this subsection, some of the layouts and its interactions, as well as the restrictions of the actions, will be exposed with some graphical representations made with Figma[6] by the designers of the company. The rest will be exposed while explaining the implementation and its functionality. Figure 5 shows what the user expects to see when accessing



Mostrando 4 Plantillas de instalación

Punto recarga coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec eget tellus non erat feugiat pulvinar.
Hace 1 mes • Personalizada

Aerotermia en adosado ✓ Activo
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec eget tellus non erat feugiat pulvinar.
Hace 1 mes • Aerotermia

Placas solares Archivado
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec eget tellus non erat feugiat pulvinar.
Hace 1 mes • Fotovoltaica

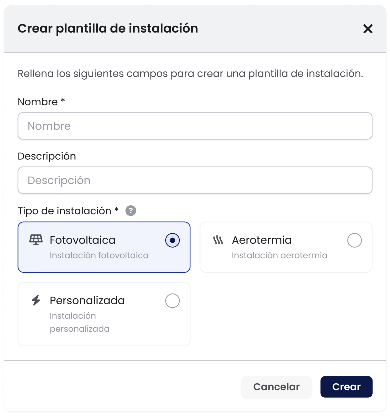
Fotovoltaica ✓ Activo
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec eget tellus non erat feugiat pulvinar.
Hace 1 mes • Fotovoltaica

Fig. 5: Plant template list design

[6]Collaborative application for interface design.

the template list. The upper left hand of the header of the view is reserved for the title that counts the number of items exposed, while on the right one appears a button with which new templates are created. Just underneath the title, there is a search bar and the filters configured with ElasticSearch.

When it comes to the items, the text in bold represents the template name; while the icon beside represents its status, which is changeable with the dropdown options shown once clicked the icon at the end of the item card. The text shown below the title is the optional description. At. the bottom, there is some extra information: the creation date and the type.



Crear plantilla de instalación ✕

Rellena los siguientes campos para crear una plantilla de instalación.

Nombre *
Nombre

Descripción
Descripción

Tipo de instalación *
● Fotovoltaica — Instalación fotovoltaica
○ Aerotermia — Instalación aerotermia
○ Personalizada — Instalación personalizada

Cancelar | Crear

Fig. 6: Plant template create view

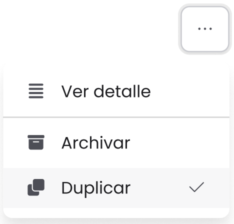


Fig. 7: Active status drop down menu

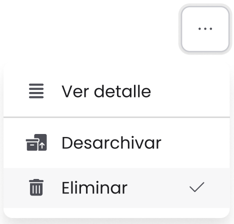


Fig. 8: Archived status drop down menu

Figure 6 exposes the pop-up that would appear once the button in the header of Figure 5 is clicked. This dialog has a form where the name field is compulsory, while the description is not. Radio options represent the type, meaning that only one of the three can be selected at a time. It can be concluded that the fields marked with an asterisk are required ones.

Each template has its actions, which are shown once clicked the button at the end of the item card. It opens the drop down menus shown in Figure 7 or 8 depending on the status of the template. These actions allow users to change the status, access the detail (although, if the user clicks the title of the template it has the same interaction) or duplicate.

It has to be said that, the name must be unique, so if the user enters a name that already exists the form will be invalid; thus, the "create" button must be disabled while an error message should appear at the button of the input field. Due to this restriction, when trying to publish an archived template, it should check whether there already exists one with that name. When it comes to duplicating, the name of the newly created one will be slightly different.

8 Implementation

In this section, the main process interactions of the project will be exposed.

8.1 Template list and CRUD operations

As previously mentioned, some of the requirements were related to creating, updating (which are called CRUD operations) and the correct visualization of the templates.

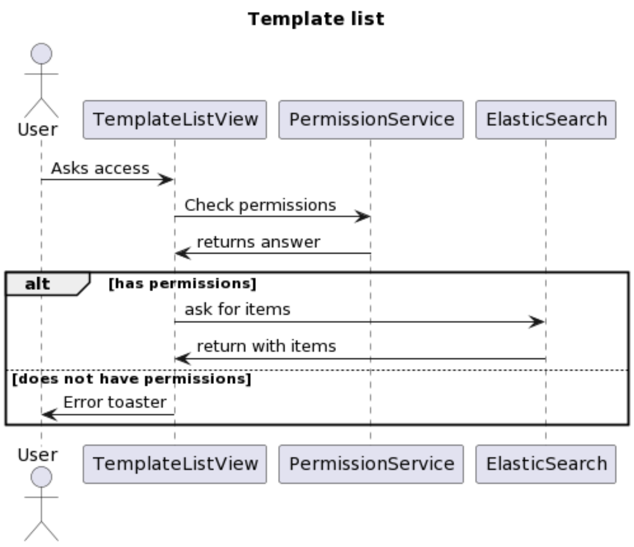


Fig. 9: Template list sequence diagram

The Figure above shows the steps that the program performs when the user wants to access the template list. There is a clear distinction between the front end classes and back end ones: only the service and the view belong to the former group.

Once the enter link is clicked, the angular routing system asks the template list class for the HTML object containing the view. Before returning the results, it asks the permission service whether the querying user has permission to do these actions or not. If positive, then the front end emits a search petition to ES specifying the parameters wanted. Only items that fulfil the requirements are returned. The reason why it is asked to ES and not to the database is that, as mentioned previously, the company handles the large lists by storing a copy in the search engine to speed up the querying process and further filtering if the user requires it.

As the condition shown in Figure 9, when the user does not have access permissions, a toaster is shown to notify the user about the problem. Not only permissions issues trigger this toaster, but any problem in the process; for example, establishing the connection with ES, is also alerted to the user by using toasters.

Once the user has accessed the template list, the "create" button will be displayed. The creation process starts when the user clicks this button, which should show a pop-up with the creation form. However, this pop-up requires knowing the solution engine allowed depending on the domain the user is in. The domain can be understood as the version of the platform, it may indicate the type of system subscription that the user has. Some users, instead of subscribing,

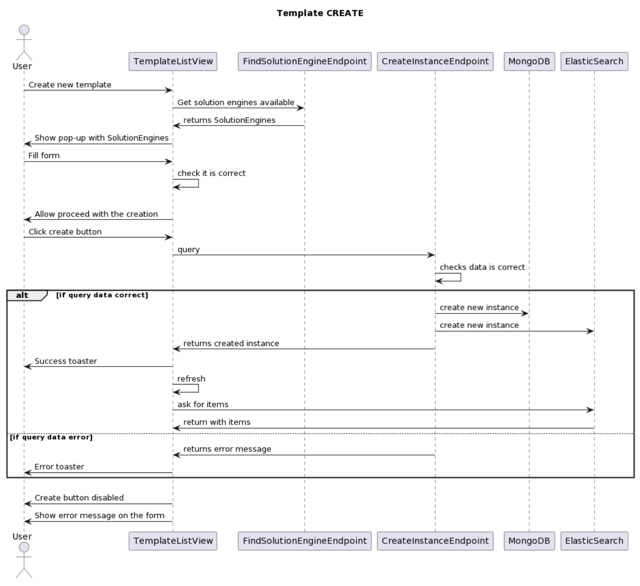


Fig. 10: Template create sequence diagram

have their own custom domain. Thus, the template list view queries the solution engines allowed and displays it in form of radio selectors in the dialog, as shown in Figure 6.

When the user has filled out the form, the correctness of the data is checked, if it is right, then the template list class enables the create button. Otherwise, the reason for the error is notified in the pertinent field.

When the user confirms the creation, a query is sent to the endpoint in charge of generating new templates. This API reevaluates the data received, if it is negative, it returns an error message to the front end, and the template list catches the failure, showing a toaster with the problem.

Whereas, when positive, it creates the instance in the database, copies the files specified in the template model to ElasticSearch, and returns the new template to the front end.

At this point, the template list reloads itself and asks the items to show to ElasticSearch.

8.2 Template detail view

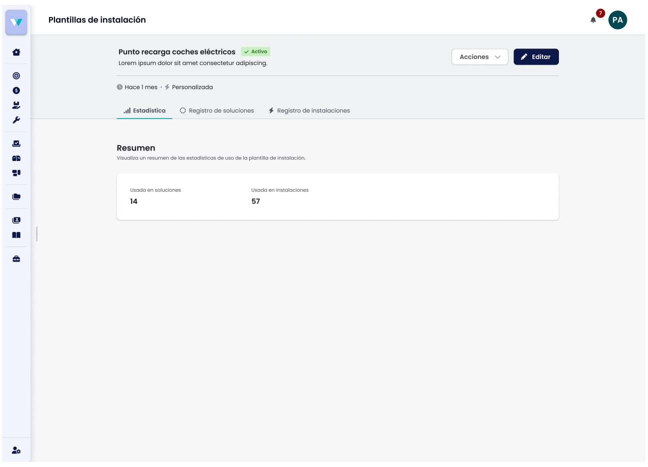
In this part, the detail view of the template will be explained. It includes the configuration of it, the assignment of several relations.

As exposed in Figure 11, the page header includes action buttons that facilitate template editing and configuration of its status, which is similar to the capabilities available in the template list. Additionally, it displays pertinent information about the template, including its name, description, and status.

8.2.1 Statistics

This part satisfies the requirement of ”facilitating the retrieval of statistical data related to template usage.” It needs the development of an endpoint that interacts with the database and retrieves information from the solution and corresponding plant tables based on the template's system type.

On the front end, a dedicated tab will be available within the detailed view, showing the data fetched by invoking the mentioned endpoint, as shown in the Figure 11. This functionality is particularly useful to users as it shows them the popularity of the created templates.



Punto recarga coches eléctricos

Lorem ipsum dolor sit amet consectetur adipiscing.

Hace 1 mes · Personalizada

Resumen

Visualiza un resumen de las estadisticas de uso de la plantilla de instalación.

Usada en soluciones 14

Usada en instalaciones 57

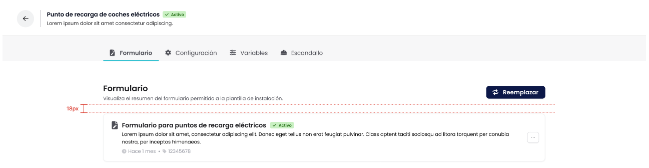
Fig. 11: Template statistics screen

8.2.2 Form template

Different types of installations utilize distinct design methods. For photovoltaic systems, a graphical interface is employed, enabling users to draw panels on a map, specify their inclination, select the desired number of panels, and access other related features. In the case of aerothermal systems, a specific form must be completed, providing the necessary data to configure the installation. However, personalized installations present a challenge. Since the usage of such open system types varies and lacks a fixed approach, a flexible and customisable design interface is required: custom forms.

To improve the user experience, the platform offers system default form templates, and users can also choose from their own pre-existing form templates. This needs establishing a relationship between the form template model and the plant template model, where each plant template can be assigned a form template. Consequently, the form template needs to be incorporated into the plant template model in the back end. While in the front end, a tab should be added to display available form templates, allowing users to select and assign them as shown in Figure 12. Additionally, options for modifying or deleting the assigned form should be provided. These actions require updating the database and should make use of the previously created update endpoint. This view requires an additional component that represents the form's empty state, as illustrated in Figure 18 in the Appendix. It is displayed when the template lacks any assigned form template. To improve user experience, the word ”Add” within the displayed text is interactive and works similarly to the button. Clicking on it triggers a popup display, following the graphical representation shown in Figure 13.

The dialog box that appears requests a title for the form that will be related, which will correspond to the name for the design tab of the installation that applies the template. Therefore, this input is required and if it is not filled, the add button is disabled Similar behavior applies to the form selector, which is an autocomplete input.

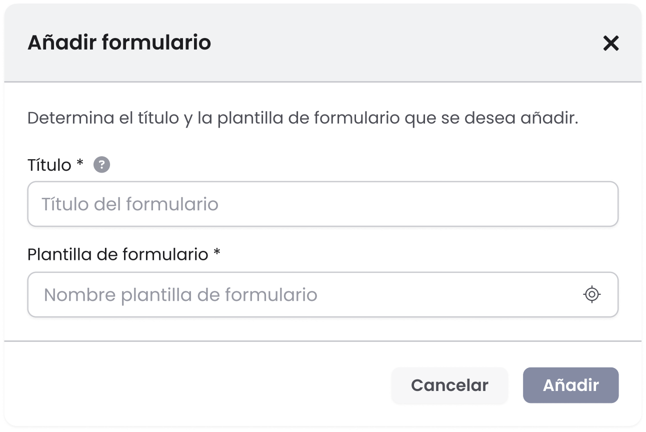


Punto de recarga de coches eléctricos Activo
Lorem ipsum dolor sit amet consectetur adipiscing.

Formulario
Visualiza el resumen del formulario permitido a la plantilla de instalación.

Formulario para puntos de recarga eléctricos Activo
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec eget tellus non erat feugiat pulvinar. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.
Hace 1 mes • 12345678

Fig. 12: Form template filled



Determina el título y la plantilla de formulario que se desea añadir.

Título *
Título del formulario

Plantilla de formulario *
Nombre plantilla de formulario

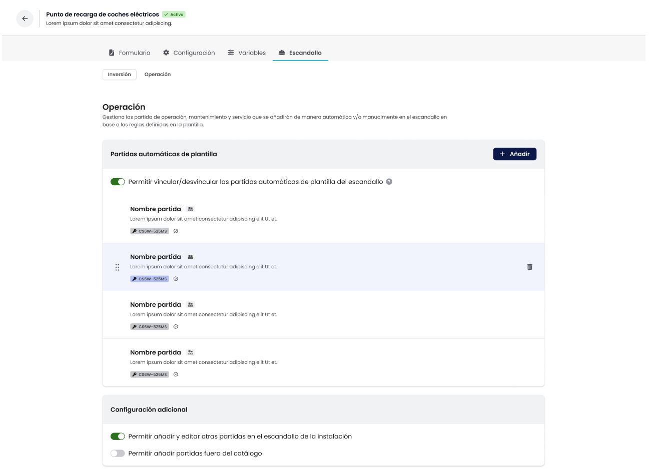
Fig. 13: Add form template pop up

The autocomplete input is a distinct component that functions as a selector, enabling users to choose from multiple options while also serving as an input field. The entered text serves as a query parameter for filtering the names of form templates stored in ElasticSearch. The outcome is presented as a list of selectable options within the selector.

8.2.3 Bill of materials

The bill of materials serves as a data object that contains various product types associated with the template. It provides the user with the ability to define default products, distinguishing between those related to investment and operational aspects. These defaults products are automatically assigned to the installations that uses the template. In the detail view of the installation, it can be found out a tab with the same product table as the below.

To present these two distinct product lists, the interface employs separate tabs, as shown in the figure below. Additionally, configuration options in the form of toggles are available. These toggles facilitate the disassociation of products from the template’s bill of materials and enable the inclusion of alternative products from the installation’s bill of materials tab. Notably, the latter toggle only becomes accessible when activated, indicating the user’s ability to personalize products outside the predefined product list.



Punto de recarga de coches eléctricos Activo
Lorem ipsum dolor sit amet consectetur adipiscing.

Operación
Gestiona las partida de operación, mantenimiento y servicio que se añadirán de manera automática y/o manualmente en el escandallo en base a las reglas definidas en la plantilla.

Partidas automáticas de plantilla

Permitir vincular/desvincular las partidas automáticas de plantilla del escandallo

Nombre partida
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
CS6W-525MS

Nombre partida
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
CS6W-525MS

Nombre partida
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
CS6W-525MS

Nombre partida
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
CS6W-525MS

Configuración adicional

Permitir añadir y editar otras partidas en el escandallo de la instalación

Permitir añadir partidas fuera del catálogo

Fig. 14: Bill of materials

8.3 Relation between form template, plant template and bill of materials

In Figure 14, the user has the option to add a product to either the inversion material list or the operation material list. This section focuses on the back end perspective of the task. The user can customize the addition of a product, rather than making it a default for all installations. They can define rules regarding the quantity and whether the product should be added.

These rules are influenced by two factors: the type of installation and the related form template, if applicable. Thus, the user can create rules associated with form template fields or constants related to specific installation types. For example, the user can restrict a product to be added only to installations where the form template’s ”How many square meters has the flat” field is greater than m^2.

To achieve this, the product selection view needs to call the backend endpoint generated in this section. This endpoint reads the related form template and retrieves the types of fields it contains, such as selectors, text inputs, radio groups, and text areas. Along with the constants dependent on installation types, these elements form all the possible variable restrictions. The endpoint then returns this object to the frontend, which processes and displays it as a selector. Depending on the user’s selection, operational actions (e.g., ”is greater than,” ”is equal to,” ”is contained in”) and a selector to choose the corresponding option, or a text input for free-form text, are presented.

8.4 Populating script

Given that the platform is currently operational with existing user data stored in the database, releasing this project as it is would lead to problems. Errors would occur when the plant template cannot be found in the current solution.

To address this problem, considering that the company makes use of the non-relational database MongoDB, a JavaScript script needs to be created for populating purposes. The primary aim of this script is to break the connection between the solution and the plants, as well as between the solution and the solution engine. This can be achieved through two steps.

Firstly, it is necessary to determine the type of installation the solution is associated with. This information can be obtained by locating the solution engine linked to the solution. With this information, a new template can be generated, containing the bill of materials currently present in the solution. The template ID should then be stored in both the solution and the plants associated with that solution.

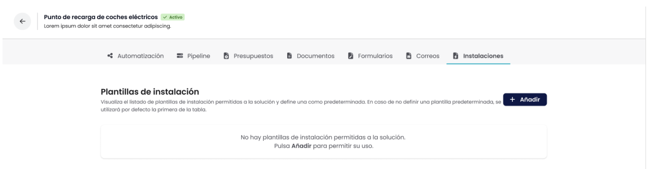
Subsequently, the solution ID of the plants, as well as the solution engine and BOMTemplate (which is the model that represents a bill of materials) of the solutions, can be cleared. This will effectively eliminate these relationships.

However, it is important to note that executing this script at the beginning of the project would cause the platform to cease functioning. Therefore, the timing for executing the script needs to be carefully chosen.

8.5 Solution - plant template relation

In order to establish a connection between these two models in the context of the front end, it is necessary to incorporate a means of establishing this relationship within the detailed view of a solution.

As previously indicated, the solution contained a bill of materials, which was displayed in the solution detail according to the specific type of installation. However, with the introduction of the new approach within this project, the solution will no longer have a directly associated bill of materials. Instead, it will have the capability to encompass one or multiple plant templates through a specific tab within the solution. Since the plant templates defines the solution engine, it is no longer needed when creating a new solution; so it should be removed from its creation pop up.


Punto de recarga de coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet consectetur adipiscing.

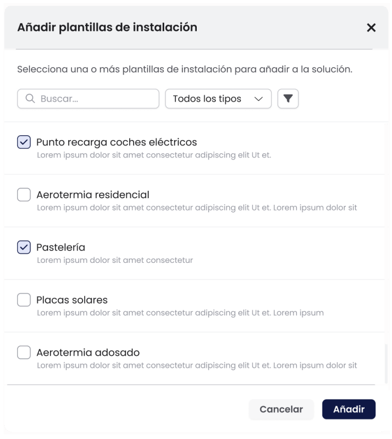
Plantillas de instalación
Visualiza el listado de plantillas de instalación permitidas a la solución y define una como predeterminada. En caso de no definir una plantilla predeterminada, se utilizará por defecto la primera de la tabla.

No hay plantillas de instalación permitidas a la solución.
Pulsa **Añadir** para permitir su uso.

Fig. 15: Plant template tab in solution detail view

Figure 15 exposes the appearance of the plant template configuration tab when the solution does not contain any plant template-related elements. By selecting either the explicit button located at the upper right corner or the ”Add” word highlighted in bold within the box, a pop-up window, as depicted in Figure 16, will be displayed.

This dialog presents a list of all existing plant templates. The list can be filtered using the search bar or the selector, which offers four options: displaying all types of templates, exclusively displaying aerothermal templates, exclusively displaying power plant templates, or exclusively displaying custom templates. The search bar filters the templates based on their name and description. These filtering mechanisms act over ElasticSearch, which serves as the source for extracting the relevant data.


Añadir plantillas de instalación

Selecciona una o más plantillas de instalación para añadir a la solución.

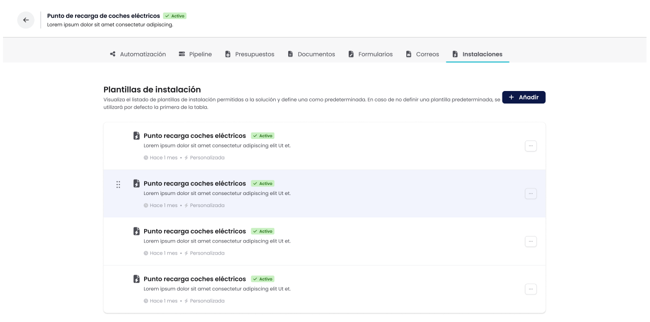
- [✓] Punto recarga coches eléctricos
  Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
- [ ] Aerotermia residencial
  Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et. Lorem ipsum dolor sit
- [✓] Pasteleria
  Lorem ipsum dolor sit amet consectetur
- [ ] Placas solares
  Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et. Lorem ipsum
- [ ] Aerotermia adosado
  Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et. Lorem ipsum dolor sit

Fig. 16: Pop up to add plant template to solution

The plant templates included in the solution will be available to the user when creating a process instance using this solution; if empty, it will show the panel exposed in Figure 15 in the Appendix. The order in which they appear will correspond to the list provided, highlighting the importance of the drag-and-drop functionality demonstrated in the emphasized instance shown in Figure 17. The dots on the left indicate draggable components, allowing the user to rearrange the items and select their desired order, thereby facilitating the future selection of popular options.

Furthermore, when the right button of an instance is clicked, a drop-down option list is displayed. This list allows the user to delete the plant template, view detailed information about the template (which will redirect the user to the corresponding URL), and set or unset a plant template as the default. Whenever there is a default option, the display changes to the one illustrated in Figure 19

If the solution includes a default plant template, the system will automatically create a plant using this default template when the user creates a process instance (modifiable if wanted). On the other hand, if there is no default plant template, the system will not generate any plant directly, leaving it up to the user to create one manually. In this case, the user can choose from the plant templates provided by the solution, if any are available.


Punto de recarga de coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet consectetur adipiscing.

Plantillas de instalación
Visualiza el listado de plantillas de instalación permitidas a la solución y define una como predeterminada. En caso de no definir una plantilla predeterminada, se utilizará por defecto la primera de la tabla.

Punto recarga coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Punto recarga coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Punto recarga coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Punto recarga coches eléctricos ✓ Activo
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Fig. 17: Solution’s plant template list

8.6 Back end modifications

This section represents the primary objective of the project, which consists at modifying the solution entity, erasing some of its relations such as the bill of materials, solution engine, and other data that have been relocated to the plant template. As well as breaking the dependency between solution and plants, replacing it for plant template. This implies changing all the endpoints, functions, utilities, models that rely on the data erased from the solution model.

In the context of LoopBack, a model represents a data entity, defining its structure, behavior, and interactions within the application. LoopBack offers a wide variety of features and utilities for effectively working with models, including automated API generation, data validation, access control, and remote methods. The model defines the data that will be persisted in Elastic search. Consequently, if the solution and its solution engine must be removed from the power plant, they must also be eliminated from the corresponding model. The same applies to other entities requiring modifications, such as the client applications, budgets, and aerothermal plants.

Moreover, there exist certain functions referred to as "before-save." As the name implies, these functions execute just before data is stored in the database. They are commonly employed to ensure data integrity and consistency within a model and across different models. Previously, the bill of materials instance associated with the solution required verification prior to storage in the database. However, this section of code should now be migrated to the newly created function plant-template-before-save.

Another set of tasks involves making changes to certain endpoints that depend on the relations within the solution. The modification process involves switching the model from which the data is retrieved, while preserving consistency. For instance, during the creation of a processInstance, configuration data was retrieved from the assigned solution. However, it should now verify whether the solution possesses a default plant template and obtain the relevant information from that template.

Additionally, it is important to examine the utility functions, which are used in various locations and subsequently extracted into a central "util" to eliminate redundant code. Similarly, constants, which represent shared variables across multiple models.

9 Obstacles and difficulties

The execution of the project encountered difficulties due to the inclusion of external tasks within the sprints, which impeded strict adherence to the expected timeline. Additionally, from a personal perspective, the lack of time further compounded the challenges faced.

As previously mentioned, the organization follows the Scrum methodology, employing three-week sprints. Consequently, the delays incurred must be accounted for in terms of sprints. The inclusion of these external tasks resulted in an unforeseen increase in workload, leading to a minimum delay of two sprints, preventing the project from being completed within the designated time frame.

It should be noted that since the project is being conducted within a company, meeting the deadline for the thesis does not signify the conclusion of the overall project. Even after the completion of the thesis, the project will continue until all requirements have been fulfilled.

10 Conclusion

As it has been explained in section 9, only part of the project has been finished on time; however, the skeleton of the project is considered to be almost finished. The Table 3 in the Appendix provides a concise overview of the requirements and the corresponding sections in which they are addressed and explained.

The remaining functional requirements have been designated for subsequent sprints and are therefore outside the scope of the thesis. Regarding the non-functional requirement pertaining to platform responsiveness, it has been addressed throughout each task, thus can be considered as completed. However, non-functional tasks 1 and 3 have been postponed intentionally, as the styling aspect is considered "secondary" in comparison to functionality, and the table display of the data has not yet been defined.

Acknowledgments

I would like to thank Wattwin in general for their support and for allowing me to carry out my final degree project at the company. In particular, I would like to express my gratitude to Jose María Malaguilla, Roger Física, and Meritxell Font for their tremendous assistance and patience.

References

[1] What is BPM? (2022, Feb 2) [Online]. Available: https://www.sydle.com/blog/bpm-60f88aeab250375797c93ee7

[2] Scrum – what is is, how it works, and why it's awesome [Online]. Available: https://www.atlassian.com/agile/scrum

[3] What are Gantt charts? [Online]. Available: https://www.atlassian.com/agile/project-management/gantt-chart

[4] Angular [Online]. Available: https://angular.io/

[5] TypeScript (2023, Mar 6) [Online]. Available: https://en.wikipedia.org/wiki/TypeScript

[6] LoopBack [Online]. Available: https://loopback.io/

[7] Node.js [Online]. Available: https://nodejs.org/en/about/

[8] NoSQL (2022, Nov 26) [Online]. Available: https://en.wikipedia.org/wiki/NoSQL

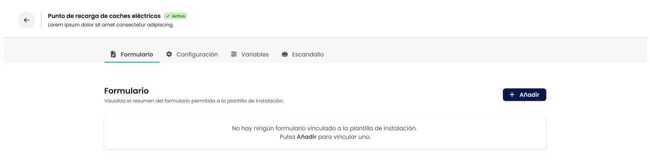
APPENDIX

A.1 Additional Tables

TABLE 3: ACCOMPLISHED REQUIREMENTS

SECTION	Functional	Non-functional
7.1 Template list and crud operations	1,2,9,10	
7.2 Template detail view		
7.2.1 Statistics	3	2
7.2.2 Form template		2
7.2.3 Bill of materials		2
7.3 Relation between form template, plant template and bill of materials	5,6	
7.4 Populating script	8	
7.5 Solution-plant template relation	13,14	
7.6 Back end modifications	4	

A.2 Additional images

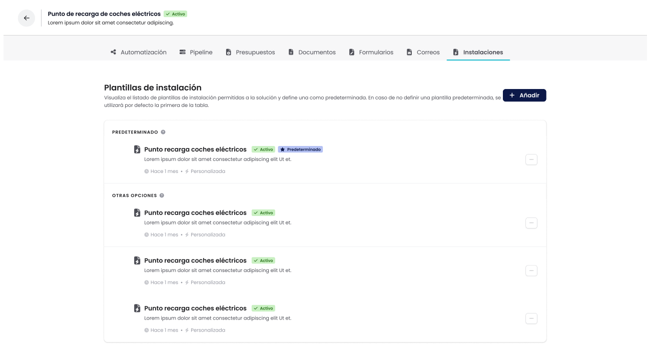


Punto de recarga de coches eléctricos
Lorem ipsum dolor sit amet consectetur adipiscing.

Formulario
Visualiza el resumen del formulario permitido a la plantilla de instalación.

No hay ningún formulario vinculado a la plantilla de instalación.
Pulsa **Añadir** para vincular uno.

Fig. 18: Form template empty



Punto de recarga de coches eléctricos
Lorem ipsum dolor sit amet consectetur adipiscing.

Plantillas de instalación
Visualiza el listado de plantillas de instalación permitidas a la solución y define una como predeterminada. En caso de no definir una plantilla predeterminada, se utilizará por defecto la primera de la tabla.

PREDETERMINADO

Punto recarga coches eléctricos
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

OTRAS OPCIONES

Punto recarga coches eléctricos
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Punto recarga coches eléctricos
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Punto recarga coches eléctricos
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut et.
Hace 1 mes • Personalizada

Fig. 19: Solution's plant template list with default one