UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DISEÑO Y DESARROLLO DE UNA APLICACIÓN WEB B2B PARA EMPRESAS DEL SECTOR DE ENERGÍA SOLAR

Diego Guerra Orozco
Tutor: Alberto Cortés Villena
Ponente: Ruth Cobos Pérez

JULIO 2016

DISEÑO Y DESARROLLO DE UNA APLICACIÓN WEB B2B PARA EMPRESAS DEL SECTOR DE ENERGÍA SOLAR

AUTOR: Diego Guerra Orozco

TUTOR: Alberto Cortés Villena

PONENTE: Ruth Cobos Pérez

Dpto. de Ingeniería del Software

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Julio de 2016

Resumen (castellano)

Ezzing Solar es una empresa de desarrollo de software en el sector de la energía solar, que desarrolla herramientas para eléctricas, fabricantes, distribuidores e instaladores del sector.

El objetivo de esta propuesta es diseñar la arquitectura, definir la forma de despliegue e implementar una aplicación web que será una plataforma configurable para diferentes proyectos de la empresa. Esta plataforma deberá comprender un CRM (administración de la relación con los clientes), gestión de productos y listas de precios, configuradores de instalaciones solares, etc.

Parte del trabajo ha consistido en evaluar y elegir las herramientas y tecnologías adecuadas a utilizar durante el desarrollo de la aplicación.

Palabras clave (castellano)

Aplicación Web, Energía Solar, Arquitectura de Sistemas, Computación en la Nube.

Abstract (english)

Ezzing Solar is a software development company in the Solar energy industry, developing tools for utilities, manufacturers, distributors and installer companies.

The goal of this proposal is to design the architecture, deployment process and implementation of a web-based application which will be a configurable platform for different projects developed by Ezzing. This platform will include a Customer Relationship Management module (CRM), products management with price lists, solar installations configurators, etc.

Part of the proposal has been to assess and choose the appropriate tools and technologies to use during the application's development.

Keywords (english)

Web Application, Solar Energy, Systems Architecture, Cloud Computing.

Agradecimientos

Quiero agradecer a los profesores de la EPS todo lo que me han enseñado durante estos años de trabajo; en especial a Ruth Cobos, por introducirme en el mundo del desarrollo web, por hacerme sitio para poder hacer el TFG con ella y ayudarme en todo el proceso, y por tratarnos a todos los alumnos con una cercanía que ahora se echa de menos.

A mis compañeros de clase, en especial a mi compañero de prácticas Javier Gutiérrez, por lo bien que lo hemos pasado, y lo duro que hemos trabajado juntos.

A Alberto Cortés, por ofrecerme mi primer trabajo como informático, confiar en mí, y ayudarme a crecer como profesional y como persona.

Al resto de compañeros de Ezzing, por el buen ambiente que hemos conseguido crear.

Y por último, a mis padres y a mi hermana, por estar siempre ahí y apoyarme en todo. Muchas gracias.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

- **API** (*Application Programming Interface*). Es un conjunto de rutinas y protocolos que especifica cómo se comunica un componente de software.
- **JSON** (*JavaScript Object Notation*). Lenguaje ligero de intercambio de datos, basado en un subconjunto del lenguaje de programación JavaScript. Ha ganado popularidad por su facilidad tanto de leerlo por humanos como de analizarlo y generarlo por programas.
- **SLA (***Service Level Agreement*). Es un contrato entre un proveedor de servicios y el usuario final en el que se define el nivel de servicio que se puede esperar del proveedor. Esto normalmente englobará una descripción de los servicios, su disponibilidad y duración de los cortes que podrían darse, el procedimiento de reportes de problemas, y las consecuencias de no cumplir esos requisitos definidos.
- **IDE** (*Integrated Development Environment*). Entorno de desarrollo para facilitar el desarrollo en uno o varios lenguajes que soporte. Suelen estar comprendidos por un editor de texto, herramientas de depuración y automatización, y autocompletado de código.
- **SQL** (*Structured Query Language*). Es un lenguaje de programación diseñado para administrar datos de un gestor de bases de datos relacionales.
- **NoSQL** (*Non SQL* o *Not Only SQL*). Este término se utiliza para referirse a una base de datos que no sigue un modelo relacional. En ocasiones también se las llama *Not Only SQL* para enfatizar que pueden soportar lenguajes similares a SQL.
- **Caché**. Un sistema de caché es un componente que almacena información para que futuras consultas a ella se puedan resolver más rápidamente. La información guardada en caché puede un cálculo realizado previamente, o información duplicada más accesible que la original.
- **Framework**. Un framework es una abstracción que aporta funcionalidad genérica y de apoyo reutilizable en diferentes aplicaciones, permitiendo al usuario construir una plataforma de software más grande sobre esa base y trabajar a más alto nivel.
- **Proxy**. Un servidor proxy es un sistema o aplicación que actúa como intermediario para consultas de clientes a otros servidores.
- **MVC** (*Model View Controller*). Modelo-Vista-Controlador es un patrón de diseño para implementar interfaces de usuario, que divide una aplicación en tres partes interconectadas entre sí: el modelo administra los datos y lógica de la aplicación, la vista representa la información y el controlador manipula el modelo y la vista.
- **TDD** (*Test Driven Development*). El desarrollo orientado a pruebas es un proceso de desarrollo de software en el que, iterando en ciclos cortos, se desarrollan primero pruebas necesarias para cumplir con unos requisitos definidos, y a continuación es cuando se desarrolla el código funcional, que deberá cumplir esas pruebas.
- **CDN** (*Content Delivery Network*). Una Red de Distribución de Contenido es un sistema de servidores distribuidos que replica contenido en diferentes servidores para servírselo a un usuario basándose en su localización geográfica, permitiendo llegar a más usuarios y disminuyendo el tiempo de entrega.

1 INTRODUCCIÓN

En esta sección se expone la motivación del proyecto, y se detallan los objetivos a cumplir en las diferentes fases que lo comprenden; el resultado final del proyecto es la implementación de una aplicación web que servirá de base reutilizable sobre la que construir diferentes plataformas con ciertas funcionalidades similares de interés para la empresa de desarrollo de software Ezzing Solar.

1.1 MOTIVACIÓN

Ezzing Solar es una empresa de desarrollo de software, que desarrolla herramientas para eléctricas, fabricantes, distribuidores e instaladores del sector de la energía solar.

Su principal objetivo es proveer una plataforma en la nube donde centralizar la actividad de todos sus clientes. Sin embargo, algunos de ellos prefieren tener una plataforma específica sólo para ellos con solo una parte de toda la funcionalidad que ofrece Ezzing, o con alguna funcionalidad específica para ellos que no esté en la plataforma principal. En estos casos, Ezzing ofrece la posibilidad de desarrollar para ellos una aplicación cerrada para ellos, o bien alojada y mantenida por Ezzing, o bien alojada por el cliente en sus propia infraestructura.

Por ello, es necesario crear un entorno sobre el que poder desarrollar para diferentes plataformas, que tenga toda la funcionalidad común y reutilizable entre plataformas, pero que sea lo suficientemente configurable para poderse personalizar para cada cliente. Este entorno base también tiene que poderse configurar para integrarse con APIs existentes de Ezzing y sus herramientas de monitorización, proveer un sistema de internacionalización, solucionar de forma automatizada la generación de la estructura de la base de datos, el sistema de backups, etc.

1.2 OBJETIVOS

El objetivo de este proyecto es diseñar la arquitectura apropiada a emplear en una plataforma web con ciertas funcionalidades predefinidas, realizar su implementación, establecer el proceso de provisionamiento de los servidores involucrados y el proceso de despliegue de la aplicación.

Para ello, se ha investigado las tecnologías disponibles actualmente para los requisitos existentes: qué tipos de base de datos será más apropiada, cuáles son los servidores de almacenamiento en caché existentes y cuáles son las ventajas y desventajas de cada uno o qué frameworks y librerías existen para facilitar y acelerar el tiempo de desarrollo.

1.3 ORGANIZACIÓN DE LA MEMORIA

La memoria consta de los siguientes apartados principales:

El **capítulo 2** hace un estudio del estado del arte de las diferentes tecnologías para el desarrollo web, y se valoran los lenguajes de programación más adecuados, el tipo de base de datos que utilizar, y los servicios de cacheo de aplicaciones existentes. Por último, se

valora los proveedores de infraestructura más beneficiosos y se valora el impacto que tendrían en la arquitectura de la aplicación.

En el **capítulo 3** se describe los casos de uso del sistema, y se realiza un análisis de requisitos funcionales y no funcionales recabados.

El **capítulo 4** describe el proceso de desarrollo del proyecto que se ha seguido, y se comenta las decisiones tomadas sobre las herramientas utilizadas, detalles de implementación y la arquitectura diseñada.

En el **capítulo 5** se detalla la estructura final del software del proyecto.

El **capítulo 6** describe las pruebas realizadas durante el desarrollo.

Finalmente, en el **capítulo 7** se presentan las conclusiones del proyecto, y las mejoras que sería interesante realizar como trabajo futuro.

2 ESTADO DEL ARTE

En esta sección se ha realizado un análisis de las tecnologías disponibles, teniendo en cuenta las diferentes necesidades de la plataforma. Las decisiones tomadas finalmente se exponen más adelante, en el apartado de Diseño.

Ezzing Solar se centra en el desarrollo de aplicaciones web, al ser éste un medio a través del cual se puede llegar a usuarios utilizando cualquier tipo de dispositivo sin necesidad de descargar software específico, ni desarrollar diferentes versiones en función de su Sistema Operativo. Otras ventajas de desarrollar en el medio Web es la amplia variedad de herramientas y lenguajes de programación disponibles entre los que elegir en función de las necesidades de cada proyecto, así como la ventaja de lanzar nuevas versiones y actualizaciones de un producto y que esté instantáneamente en uso por todos los usuarios.

El objetivo de este TFG es construir una aplicación web compuesta por los servicios que se puede ver en la Figura 1.

Para evaluar los servicios que utilizar, se han investigado las tecnologías disponibles actualmente para los requisitos existentes: qué tipo de base de datos será más apropiada, cuáles son los servidores de cacheo existentes y cuáles son las ventajas y desventajas de cada uno o qué frameworks y librerías existen para facilitar y acelerar el tiempo de desarrollo.

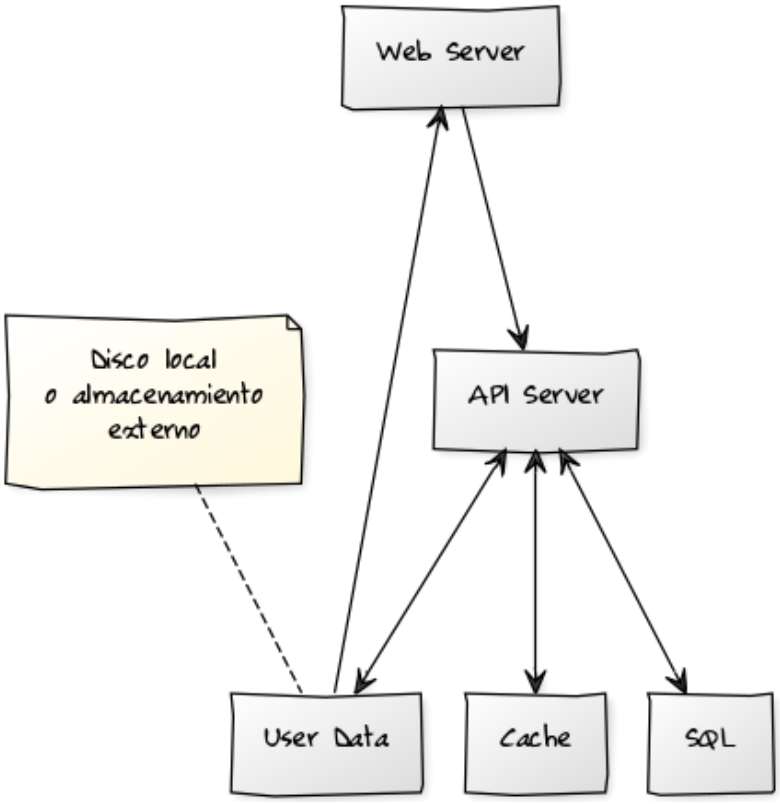


FIGURA 1. ARQUITECTURA GENERAL DEL PROYECTO

2.1 ESTUDIO DE TECNOLOGÍAS

2.1.1 BASE DE DATOS

Las bases de datos llamadas NoSQL (que no siguen un modelo relacional) han avanzado mucho en los últimos años, y tienen aplicaciones para las que son mucho más apropiadas que el modelo tradicional de base de datos relacional. En nuestro caso, la gran mayoría de información que necesitamos guardar es de carácter relacional. Por ello, es más razonable utilizar un gestor de bases de datos relacional.

Solo se han valorado alternativas de código abierto, puesto que existen productos muy maduros y avanzados, que nos permiten utilizar un producto estable, muy probado por la comunidad de código abierto, y con una gran cantidad de documentación disponible en Internet. Estos productos a menudo se pueden utilizar sin necesidad de adquirir licencias para su uso, lo que supone un coste de desarrollo menor sin perder funcionalidad que puedan ofrecer otras soluciones de pago.

Las principales alternativas de código abierto y gratuitas que se han considerado son:

2.1.1.1 MySQL

MySQL [1] es la base de datos SQL más popular y más comúnmente utilizada con fines a gran escala.

Sus principales ventajas son la facilidad inicial de uso, seguridad avanzada, un gran rendimiento sin necesidad de mucha configuración, y su capacidad de escalar. MySQL utiliza una licencia GPL, que permite usarse gratuitamente para fines comerciales, sin limitaciones. Debido a su popularidad y madurez, hay un gran número de aplicaciones de terceros, herramientas y documentación que facilitan el uso y el mantenimiento de esta base de datos.

Las principales desventajas de MySQL son su estancamiento durante los últimos años, mientras que otras alternativas han continuado evolucionando y añadiendo nuevas funcionalidades muy atractivas; y su falta de rendimiento en algunas situaciones comunes, como por ejemplo en aplicaciones con bastante actividad mixta de lectura y escritura, puesto MySQL está especialmente optimizada para soportar una gran carga de solo lectura. Además, MySQL no implementa completamente el estándar SQL. Es normal encontrarse extensiones al lenguaje SQL en las diferentes bases de datos relacionales, pero puede dar problemas inesperados utilizar un producto que no se adhiere al estándar definido.

2.1.1.2 PostgreSQL

El gestor de bases de datos PostgreSQL [2] es muy avanzado, con una gran comunidad detrás de él, cuyo principal objetivo es construir una base sólida sobre la que implementar toda la funcionalidad necesaria para una base de datos avanzada. PostgreSQL tiene como objetivo cumplir los estándares SQL, ofreciendo una gran cantidad de funcionalidad adicional . Por ejemplo, ofrece soporte para trabajar con objetos JSON e información geoespacial, metiéndose en el terreno de las bases de datos NoSQL.

PostgreSQL tiene una gran eficiencia, y está implementado de forma que resuelve muchos problemas de diseño presentes en MySQL, como por ejemplo grandes mejoras sobre cómo se gestiona la concurrencia.

Las desventajas principales de PostgreSQL son: su menor popularidad comparada con alternativas como MySQL, lo que puede suponer un menor soporte de herramientas existentes; y un menor rendimiento en ciertas situaciones, como por ejemplo en usos principalmente de lectura, en las que MySQL obtiene mejores resultados.

2.1.1.3 MariaDB

MariaDB [3] es una base de datos más reciente, que ha nacido tras al estancamiento de MySQL. Hecha por los desarrolladores originales de MySQL, ofrece la misma funcionalidad que el proyecto original, con algunas mejoras sobre ella.

En los últimos años, se ha podido ver a muchos antiguos usuarios importantes de MySQL migrar a esta nueva alternativa.

2.1.2 SERVIDOR DE CACHÉ

Utilizar un sistema de cacheo beneficia al rendimiento de una aplicación, dando un mejor aprovechamiento de los recursos disponibles, que resulta en una aplicación más rápida, con un mayor rendimiento y con un menor gasto económico.

Existen muchos tipos diferente de cacheo, que afectan a niveles diferentes de una aplicación. Los principales son:

2.1.2.1 Caché de contenido estático

Se utilizaría un CDN (Content Delivery Network) para gestionar contenido estático público, como imágenes o documentos de usuario. Existen tanto soluciones alojadas por terceros (CloudFlare [4], Amazon CloudFront [5]) como servidores a desplegar manualmente (Thumbor [6], específico para imágenes).

2.1.2.2 Cacheado de página

Para cachear páginas renderizadas, o respuestas de peticiones a una API, que son normalmente estáticas. Por ejemplo, se usaría para devolver código del frontend de la aplicación (hojas de estilos, código javascript), o el HTML final de una página, sin que esas peticiones lleguen a tocar el servidor principal. Se puede conseguir con servicios como Varnish [7], o un servidor NGINX [8] configurado como proxy con caché.

2.1.2.3 Cacheado de datos

Se utiliza internamente dentro de la aplicación para evitar hacer las mismas consultas costosas a la base de datos, o cálculos repetitivos, cacheando esos datos en su lugar. Se puede obtener de muchas formas, desde guardar esos datos en ficheros temporales para tal efecto, hasta usar servidores específicos de cacheo de objetos, como Memcached [9] o Redis [10].

2.1.3 LENGUAJE DE PROGRAMACIÓN Y HERRAMIENTAS EN EL BACKEND

Para el desarrollo del backend se ha valorado principalmente los lenguajes PHP y Python, debido a la experiencia previa de Ezzing con estos lenguajes, así como la facilidad que dan de implementar y ejecutar un servidor web basado en Linux con ellos.

Ninguno de los lenguajes tiene ninguna clara ventaja o desventaja para el desarrollo de una aplicación web, y ambos aportan funcionalidades similares y están respaldados por una gran comunidad detrás de ellos. Por ello, se ha investigado el estado de los gestores de paquetes y la disponibilidad de librerías y frameworks para ambos lenguajes.

Utilizar un framework no era absolutamente necesario, pero es una herramienta que ha ayudado a desarrollar más rápido y mejor, dando a los desarrolladores una base más robusta sobre la que implementar la lógica de negocio. En especial al implementar una plataforma de tanta complejidad, se hace necesario evaluar con detalle las alternativas disponibles.

2.1.3.1 Python

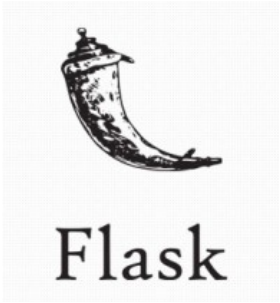
Python cuenta con el gestor de paquetes Pip [11] para buscar e instalar paquetes desarrollados por la comunidad. Es un gestor bastante bien establecido, y actualmente la convención al crear un paquete de código abierto es hacerlo disponible a través de esta herramienta.

Los principales frameworks de desarrollo web para Python son los siguientes:

2.1.3.1.1 Django

Django [12] es un framework de alto nivel que permite desarrollar aplicaciones web muy complejas, dando herramientas para implementar problemas comunes como la autenticación y autorización de usuarios, registro a través de redes sociales, internacionalización de usuarios, gestión de vistas a través de plantillas, etc. Django es un framework MVC, que da al desarrollador la estructura con la que organizar las diferentes partes que componen el proyecto.

2.1.3.1.2 Flask



Flask [13] es un microframework, más enfocado al desarrollo de APIs JSON para aplicaciones sin demasiada complejidad. En la aplicación a desarrollar podía ser una opción útil, puesto que queremos separar el backend como una API JSON.

2.1.3.2 PHP

PHP es el lenguaje que más se ha utilizado en otros productos en Ezzing, por lo que su uso facilitaría reutilizar o integrar módulos de la aplicación con otros existentes de ser necesario.

PHP cuenta con el gestor de paquetes Composer [14], con el que resulta muy fácil gestionar la autocarga de clases, la modularización de la aplicación y la importación de librerías de código abierto. Aunque Composer lleva 5 años siendo una beta y solo ha lanzado su primera versión estable en los últimos meses, hace tiempo se ha convertido en el estándar para la distribución de librerías desarrolladas por la comunidad.

PHP es un lenguaje que lleva existiendo y mejorando más de 20 años. Existen muchos frameworks disponibles para este lenguaje. Los principales a considerar debido a su madurez, su uso de la funcionalidad más moderna del lenguaje, y lo extendido de su uso en la comunidad de PHP son los siguientes:

2.1.3.2.1 Symfony

Symfony [15] es un conjunto de componentes autónomos que juntándose entre sí forman un framework con un gran abanico de funcionalidades. Sus componentes son de los más utilizados en otras librerías de PHP, y a menudo se utilizan para construir otros frameworks de PHP.

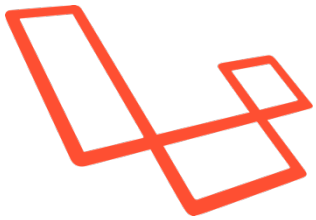
La desventaja de esta alternativa sería la necesidad de organizar correctamente el proyecto y cómo usar los diferentes componentes entre sí, a diferencia de como ocurriría con la mayoría de frameworks, donde vendría todo preparado.

2.1.3.2.2 Yii

Yii Framework [16] está enfocado en la seguridad y el alto rendimiento de la aplicación. Es un framework muy completo, en el que es muy fácil añadir extensiones de la comunidad a través de widgets y módulos.

Su principal desventaja es la poca madurez de su versión actual, Yii 2. Esta versión es una reescritura de la versión anterior en la que se ha adoptado las últimas tecnologías y las mejoras más recientes de PHP, y en ella se introducen muchos conceptos diferentes a la versión original de Yii.

2.1.3.2.3 Laravel



Actualmente Laravel [17] es el framework de PHP más utilizado. Lo sigue una comunidad muy activa, y existe una gran cantidad de plugins de terceros para añadir diferentes tipos de funcionalidad a Laravel. Es un framework tan completo como Django, y a pesar de estar en continuo desarrollo, ofrece versiones estables con soporte a largo plazo (LTS, Long Term Support), lo que resulta conveniente para realizar el mantenimiento de diferentes aplicaciones desarrolladas con Laravel.

Las principales desventajas de Laravel son su bajo rendimiento en comparación con otros frameworks similares, y la facilidad de caer en algunas malas prácticas de programación debido a la excesiva flexibilidad que puede llegar a dar.

2.1.4 LENGUAJE DE PROGRAMACIÓN Y HERRAMIENTAS EN EL FRONTEND

La capa visual de la aplicación se ha desarrollado con tecnologías diferentes, que se comunicaban con el backend a través de la API JSON con la que se exponía toda la funcionalidad.

Las tecnologías básicas que se utilizan en un navegador son los lenguajes de marcado HTML y CSS, para definir un documento y darle estilo, respectivamente. Este documento se puede editar con el lenguaje de programación JavaScript, para mostrar un contenido más dinámico, interactuar con el usuario, etc.

Cuando se quiere realizar una aplicación web en este medio, utilizar JavaScript directamente para editar el contenido de la página acaba siendo una tarea de demasiado bajo nivel y muy propenso a errores, debido a que hay que modificar manualmente el documento mostrado cada vez que la información almacenada en JavaScript cambia, y asegurarse a cada cambio de que la vista y el modelo están sincronizados. Por ello, utilizar un framework o librería que abstraiga el repintado de la vista dado un modelo es una buena idea para solucionar ese problema.

A continuación se valora varias herramientas que tienen este objetivo. Utilizan un patrón MVC (Model-View-Controller) o similar, de forma que las diferentes vistas a mostrar se muestran como plantillas, y consumiendo la API desarrollada obtendremos los modelos necesarios.

Las principales herramientas valoradas han sido las siguientes:

2.1.4.1 AngularJS

AngularJS [18] es el framework más utilizado actualmente para el desarrollo de interfaces web. Da la capacidad de utilizar plantillas, y darle dinamismo a las páginas HTML de una forma declarativa.

AngularJS es muy fácil de usar gracias a la forma en que enlaza los datos con la vista en dos direcciones; un dato puede ser modificado o

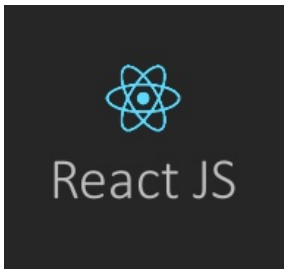
bien a través de código, o a través de la vista por el usuario, y Angular se encarga de actualizarlo automáticamente en ambas partes.

2.1.4.2 BackboneJS

Backbone [19] es más manual que AngularJS; es necesario definir cómo se relaciona la vista con los modelos de datos, y configurar cuándo se renderiza la vista para mostrar los cambios de información.

Esto resulta en un desarrollo más lento, pero puede permitir un mayor control, y una depuración más fácil, durante el desarrollo.

2.1.4.3 ReactJS



Facebook ha desarrollado ReactJS [20] como una forma novedosa de construir interfaces de usuario. Con esta librería, se enlazan los modelos de datos con la vista en una sola dirección, y se fuerza a desarrollar la interfaz de una forma más predictiva y fácil de depurar. Es una librería con unas normas muy estrictas de cómo hacer las cosas, y más difícil de aprender a usar inicialmente, pero que hace más fácil mantener y extender una aplicación una vez dominada.

React también introduce diferentes técnicas novedosas para permitir un gran rendimiento en el renderizado de la interfaz de usuario, por lo que es una alternativa muy llamativa si se trabaja con una gran cantidad de datos en el frontend.

2.1.5 Entorno de desarrollo

Dependiendo de los lenguajes de programación que se utiliza finalmente, se realizará el desarrollo con PHPStorm [21] (si el backend se desarrolla con PHP) o PyCharm [22] (si se utiliza Python). Ambos IDEs están hechos por JetBrains, una empresa reconocida por las herramientas de desarrollo que ofrece. Con ambos entornos se puede editar JavaScript y HTML, por lo que el frontend se podrá desarrollar con cualquiera de ellos.

El desarrollo y pruebas en local se realizará con la ayuda de Vagrant [23], una herramienta de gestión de máquinas virtuales para automatizar la creación de entornos de desarrollo desechables. Vagrant es una herramienta ampliamente utilizada en todos los sectores de desarrollo de software, distribuida por HashiCorp, una empresa distinguida en el sector de operaciones y administración de sistemas.

2.2 Proveedor de infraestructura

Existen diferentes proveedores de servidores en la nube que permiten a empresas sin su propia infraestructura crear y gestionar fácilmente diferentes tipos de servidores. Estos proveedores ofrecen diferentes servicios especiales de almacenamiento, cacheado o replicación que son muy atractivos y reducen el mantenimiento necesario de una aplicación. Además, con estos servicios es más fácil definir y conformarse a SLAs (Service Level Agreement) sin gastar tantos recursos en infraestructura y su mantenimiento, puesto que estos servicios ya nos ofrecen sus propios SLAs de la infraestructura que proveen, y ponen a nuestra disposición sistemas de alertas y monitorización muy robustos.

Lamentablemente, a menudo hay que adaptar la aplicación para utilizar estos servicios especiales, dificultando en el futuro moverla a otro proveedor, o a servidores físicos propios.

Idealmente la aplicación a desarrollar tiene que estar lo menos ligada posible a un proveedor de servidores, para dar la opción al cliente de gestionar el alojamiento del sistema donde ya tenga otros servicios, o en su propia infraestructura.

A continuación se listan las principales alternativas disponibles:

2.2.1 Amazon Web Services

Amazon.com fue la primera empresa que ofreció una plataforma completa de infraestructura en la nube para empresas a gran escala. Todos esos servicios están disponibles a través de la plataforma AWS [24]. Disponen de centros de datos por todo el mundo, y destacan por la gran cantidad de servicios que ofrecen. Sobre la infraestructura de AWS existen servicios tan grandes como las plataformas de Dropbox o Netflix.

A continuación se destacan los servicios más interesantes que podría aprovechar nuestra aplicación utilizando AWS:

Elastic Compute Cloud (EC2) es como llama AWS a los servidores virtuales que ofrece. Estos servidores son una máquina base con el sistema operativo que se elija, entre los que están las principales distribuciones de Linux.

Es posible agrupar diferentes máquinas EC2 en grupos dinámicos (Auto Scaling Groups), para administrar a la vez todos los servidores englobados en un grupo. Por ejemplo, se podría configurar un balanceador de carga que dirija las peticiones a las máquinas en un grupo dinámico, y monitorizar la salud del grupo de forma que si una máquina tiene mal funcionamiento automáticamente se la saque del grupo y se genere una alerta. También se puede definir acciones mucho más complejas, como darle al grupo la capacidad de aumentar o decrecer el número de máquinas en él dependiendo de la cantidad de tráfico que haya.

Otro servicio muy beneficioso sería **Amazon Relational Database Service** (RDS), un servicio de alto nivel para gestionar bases de datos relacionales tradicionales. Utilizándolo, se puede disponer de una base de datos configurada para tener alta disponibilidad, ocultándonos la dificultad de replicar la base de datos, gestionar copias de seguridad, etc.

El tercer servicio a destacar es **Amazon Simple Storage Service** (S3). Este servicio es un almacenamiento de objetos a través de una API HTTP sin límite de contenido, y que puede ser replicado a través de un CDN (**Amazon CloudFront**). Es un servicio rápido, con garantías de alta disponibilidad, y muy poco costoso.

Utilizar este servicio en lugar de un sistema de ficheros tradicional para el almacenamiento de cualquier tipo de documento e imágenes, nos evitaría tener que preocuparnos de que siempre hubiera espacio en disco, y permitiría centralizar esos documentos en un único sitio para poder alojar la aplicación en varios servidores.

2.2.2 *Google Cloud Platform*

Esta plataforma es el equivalente a AWS que lanzó Google para competir contra Amazon, y actualmente ofrece un gran abanico de servicios.

En esta plataforma, los servidores virtuales se llaman **Google Compute Engine**, y tienen funcionalidad equivalente a EC2. También existen servicios equivalentes a S3 y RDS en Amazon: **Google Cloud Storage** y **Google Cloud SQL**, respectivamente.

2.2.3 *Microsoft Azure*

Microsoft también ofrece una alternativa más con la que compite contra AWS y Google Cloud Platform. La principal desventaja de Azure es que está mucho más orientada a sistemas operativos Windows que Linux, aunque en los últimos años han mejorado mucho su oferta para Linux.

En Azure existe una también un servicio de bases de datos administrado, **Microsoft Azure SQL Database**. Este servicio parece ofrecer una base de datos basada en Transact-SQL, una extensión de SQL propietaria de Microsoft.

El servicio de almacenamiento que ofrece es **Azure Storage**. Es una alternativa mucho más compleja, que incluye funcionalidad para muchos casos de uso diferentes:

- Almacenamiento de objetos sin estructura, equivalente a S3 y Google Cloud Storage.
- Almacenamiento estructurado tablas. Funciona como una base de datos NoSQL de claves y valores.
- Almacenamiento en colas, para proveer comunicación entre componentes a través de colas.
- Almacenamiento de ficheros. Ofrece un sistema de ficheros real que utiliza el protocolo SMB, usado en muchos servicios de Microsoft.

2.3 Arquitectura

La aplicación se tiene que desplegar de forma que ofrezca alta disponibilidad. Esto se puede cumplir identificando los **Single Point of Failure** (SPOF) de la aplicación, y conformándolos a un sistema de alta disponibilidad.

Para mejorar el tiempo de respuesta y el rendimiento global de la aplicación, se introduciría una capa de caché de aplicación. Durante el desarrollo se ha identificado los puntos más utilizados con operaciones repetitivas costosas, y se han mitigado moviendo sus resultados al sistema de caché.

El código desarrollado de la aplicación se debe servir desde diferentes servidores, para facilitar un escalado horizontal de la aplicación como respuesta a un aumento de la cantidad de usuarios. Para que esto se pueda realizar, es necesario que los servidores no contengan estado de la aplicación; la información del usuario autenticado, archivo subidos o generados para él, y cualquier información cacheada deben mantenerse fuera de los servidores. De esta forma, el balanceador de carga no necesita tener lógica para redirigir a cada usuario siempre al mismo servidor.

3 ANÁLISIS DE REQUISITOS

En este apartado se analiza las diferentes funcionalidades de la aplicación a desarrollar.

Esta deducción de requisitos se ha realizado a través del análisis de proyectos reales de Ezzing Solar que se van a construir sobre este sistema, así como haciendo un estudio de las características y limitaciones de las soluciones actuales a las que reemplazará.

3.1 REQUISITOS FUNCIONALES: CASOS DE USO

3.1.1 ACTORES DEL SISTEMA

Los actores del sistema son los tipos de usuarios que interactúan con él. La plataforma diferenciará entre usuarios que pueden tener uno de los siguientes roles:

Actor	Descripción
Administrador de empresa	Usuario administrador del resto de usuarios de su empresa. En la plataforma se registrarán diferentes empresas instaladoras, que son clientes del dueño de la plataforma.
Usuario de empresa	Un usuario normal, perteneciente a una de las empresas existentes.
Administrador de plataforma	Un administrador de toda la plataforma, con visibilidad y capacidad de administrar las diferentes empresas registradas.

TABLA 1. ACTORES DEL SISTEMA

3.1.2 DIAGRAMA DE CASOS DE USO

A continuación se muestra un diagrama de casos de uso describiendo las funcionalidades principales de la aplicación.

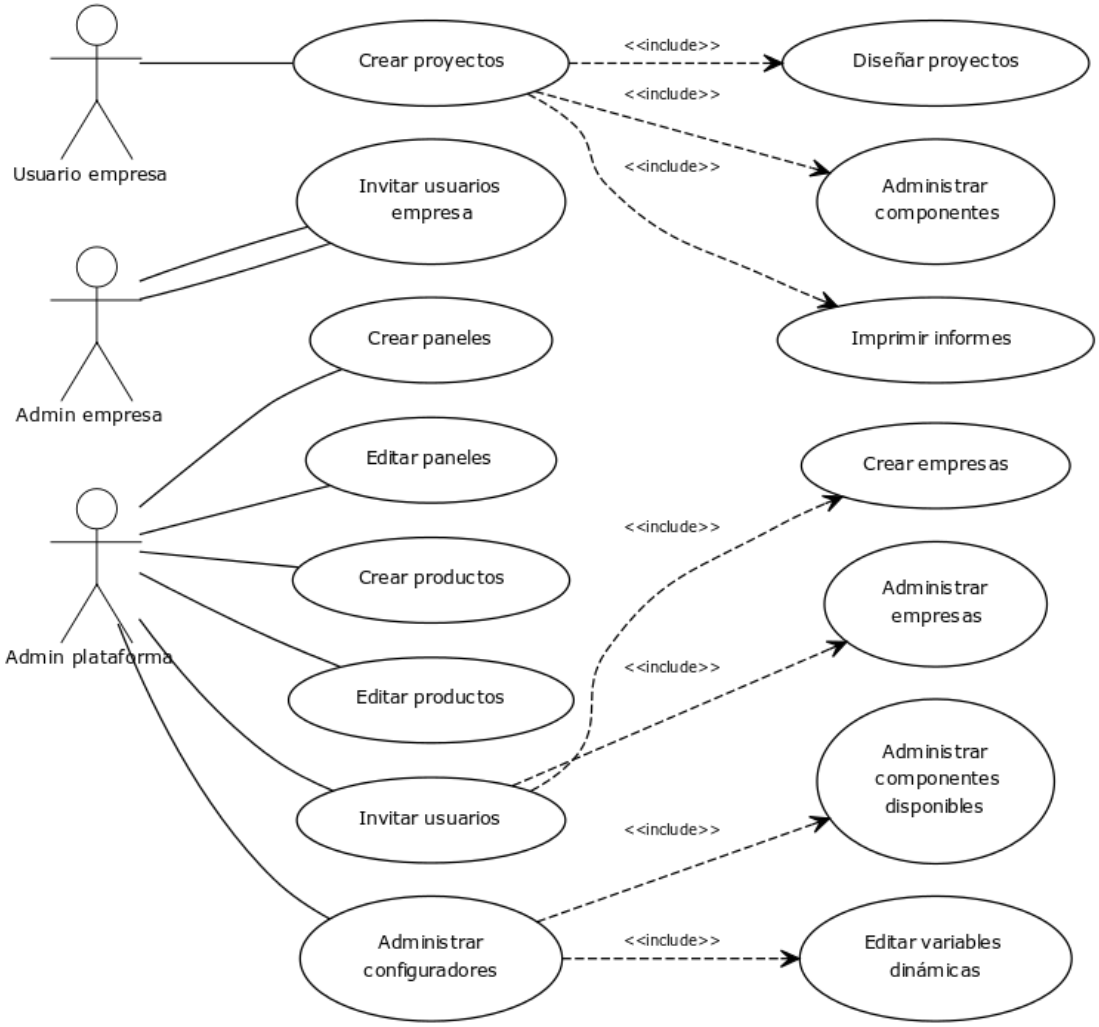


FIGURA 2. DIAGRAMA DE CASOS DE USO

3.1.3 CASOS DE USO DEL SISTEMA

Los casos de uso del sistema son los diferentes flujos de trabajo que puede realizar cada actor en la aplicación.

A continuación se detalla los diferentes casos de uso identificados, organizados por los principales subsistemas a los que pertenecen.

3.1.3.1 Registro en el sistema

CU-001: Registro en el sistema		
Actor	Usuario de empresa.	
Descripción	Un nuevo usuario deberá poder registrarse en el sistema.	
Precondición	El usuario ha abierto la página principal de registro. El correo del usuario aún no ha sido usado en la plataforma.	
Secuencia normal	1. El usuario llega a la página de acceso a la plataforma. 2. El usuario se mueve a la página de registro a través de un enlace visible en esa página. 3. El usuario introduce su nombre y apellidos, nombre de compañía y email, y pincha en el botón de registro. 4. El sistema devuelve un mensaje de éxito al usuario, indicando que se ha registrado correctamente y recibirá instrucciones para acceder en su correo electrónico. 5. El usuario accede a un mensaje de confirmación en la bandeja de entrada de su correo electrónico, y abre un link de confirmación de su correo. 6. El sistema devuelve un mensaje de éxito al usuario, lo autentica y lo redirige a la página de inicio.	
Excepciones	**Paso**	**Acción**
	2	*Si el usuario intenta autenticarse introduciendo credenciales inválidas,*
		E.1 El sistema devuelve un mensaje de autenticación errónea.
		E.2 Se cancela el caso de uso.
	3	*Si el usuario deja vacío alguno de los campos,*
		E.1 El sistema devuelve un mensaje de registro erróneo.
		E.2 Se cancela el caso de uso.
		Si el usuario introduce un email que ya existe en el sistema,
		E.1 El sistema devuelve un mensaje de registro erróneo.
		E.2 Se cancela el caso de uso.
	4	*Si el usuario introduce un email que no le pertenece,*
		E.1 El usuario nunca recibirá un correo de confirmación y no podrá acceder al sistema. El usuario dueño del correo recibirá un mensaje de confirmación con un aviso para notificar si él no se ha intentado registrar. Si decide notificar el error, el sistema eliminará al nuevo usuario.
		E.2 Se cancela el caso de uso.
	5	*Si el usuario intenta autenticarse introduciendo las credenciales con las que se ha registrado,*
		E.1 El sistema devuelve un mensaje de confirmación de email necesaria.
		E.2 Se cancela el caso de uso.

TABLA 2. CU-001 REGISTRO EN EL SISTEMA

3.1.3.2 Crear proyecto

CU-002: Crear proyecto	
Actor	Usuario de empresa.
Descripción	Un usuario autenticado deberá poder crear un proyecto para simular una instalación.
Precondición	El usuario se ha autenticado en el sistema.
Secuencia normal	1. El usuario accede a la página de creación de proyectos. 2. El usuario introduce la dirección del proyecto en una caja de búsqueda. 3. El sistema busca información detallada de la dirección del proyecto con la API de Google Maps Geocode, y se rellena diferentes campos del proyecto con ella. 4. El usuario añade manualmente información sobre el cliente del proyecto. 5. El usuario selecciona el configurador que usar en el proyecto. 6. El usuario introduce las áreas y dimensiones de la instalación. 7. El sistema presenta un listado de los componentes necesarios calculados. *8. Si el usuario quiere realizar alguna modificación,* 8.1. El usuario edita diferentes parámetros junto al listado de componentes calculado. 8.2. El usuario modifica manualmente la cantidad de componentes calculados. 9. El sistema presenta al usuario informes disponibles para descargar información del proyecto completado, o la opción de compartirlo por correo.
Excepciones	**Paso** **Acción** 3 *Si la dirección introducida por el usuario no se encuentra, o no hay información suficiente sobre ella para rellenar todos los campos requeridos,* E.1 El sistema marca los campos vacíos con un mensaje de error, para que el usuario los rellene manualmente. E.2 El usuario rellena todos los campos requeridos. E.3 El caso de uso continúa como normalmente. 6 *Si el configurador elegido no es compatible con el área introducida por el usuario,* E.1 El sistema informa de las reglas que no está cumpliendo el área. E.2 El usuario modifica las áreas de la instalación, o vuelve al paso 5 del caso de uso seleccionando un configurador diferente.

TABLA 3. CU-002 CREAR PROYECTO

3.1.3.3 Invitar usuario de empresa

CU-003: Invitar usuario de empresa	
Actor	Administrador de empresa.
Descripción	Un usuario administrador de su empresa deberá poder invitar a un nuevo usuario, que estará asociado a la empresa del administrador.
Precondición	El administrador de empresa se ha autenticado en el sistema.
Secuencia normal	1. El administrador accede a la lista de usuarios activos en su empresa. 2. El administrador pincha en el botón de invitar nuevo usuario a unirse. 3. El administrador introduce el email del nuevo usuario. 4. El sistema envía un email de invitación al nuevo usuario. 5. El nuevo usuario abre un link en el email recibido. 6. El sistema confirma el email del nuevo usuario al abrirse ese link, y muestra una página para introducir el resto de información del usuario. 7. El nuevo usuario introduce sus datos. 8. El sistema devuelve un mensaje de éxito al usuario, lo autentica y lo redirige a la página de inicio.
Excepciones	**Paso** **Acción** 3 *Si el email introducido ya existe en la plataforma,* E.1 El sistema devuelve un mensaje de registro erróneo. E.2 Se cancela el caso de uso.

TABLA 4. CU-003 INVITAR USUARIO DE EMPRESA

3.1.3.4 Administrar configurador

CU-004: Administrar configurador		
Actor	Administrador de plataforma.	
Descripción	Un administrador del sistema deberá poder administrar los configuradores existentes, editando su nombre y descripción, constantes y variables dinámicas en uso, y los componentes disponibles en el configurador.	
Precondición	El usuario se ha autenticado como administrador de la plataforma, y existen configuradores activos que editar en el sistema.	
Secuencia normal	1. El administrador accede a la página de administración de configuradores. 2. El administrador abre el configurador a editar. 3. El administrador añade y quita productos de los grupos de componentes válidos. 4. El administrador modifica variables del configurador. 5. El sistema guarda los cambios. Proyectos a partir de ese momento usarán la nueva información.	
Excepciones	**Paso**	**Acción**
	3	*Si se deja vacío un grupo de componentes que no pueda estarlo,*
		E.1 El sistema mostrará un error indicando que la modificación es incorrecta.
		E.2 Se cancela el caso de uso.
		Si se añade a un grupo de componentes disponibles un producto de un tipo inválido,
		E.1 El sistema mostrará un error mostrando los tipos de productos que se pueden añadir.
		E.2 Se cancela el caso de uso.
	4	*Si se intenta dar un valor inválido a una variable,*
		E.1 El sistema mostrará un error indicando el tipo de variable y los valores válidos que puede tener.
		E.2 Se cancela el caso de uso.

TABLA 5. CU-004 ADMINISTRAR CONFIGURADOR

3.2 Requisitos de interfaces externos

3.2.1 Interfaces de usuario

La interfaz de usuario deberá estar enfocada a ofrecer una gran usabilidad y ser sencilla, ocultando siempre que sea posible la complejidad que pueda tener una operación hasta que el usuario decida hacer operaciones de forma avanzada.

3.2.2 Interfaces hardware

El backend que sirve la API y la aplicación web deben poder desplegarse en un servidor Linux con una distribución conocida (Ubuntu, CentOS) una vez se haya instalado en ella las dependencias necesarias (PHP 5.6+, drivers cliente MySQL, MCrypt).

La instancia de MySQL que contendrá los datos de la aplicación, y el servidor Redis utilizado para la caché de datos de la aplicación se alojarán en servidores independientes, dado que la API estará replicada en diferentes servidores conectados contra ellos.

El frontend de la aplicación serán ficheros estáticos de HTML, JavaScript y CSS, por lo tanto podrá distribuirse por separado con cualquier servidor estático. Idealmente, se replicará con un CDN para minimizar el tiempo de carga inicial al abrir la aplicación.

3.2.3 Interfaces software

La aplicación debe exponer toda su funcionalidad a través de una API HTTP, utilizando JSON como formato de intercambio de datos.

Con la aplicación se provee una interfaz web que expone la totalidad de la funcionalidad de la API, pero ésta debe poder consumirse en el futuro para su utilización en otras plataformas, como aplicaciones de escritorio o aplicaciones nativas en Android o iOS.

3.3 Requisitos de rendimiento

El consumo de recursos de la aplicación debe ser lo más bajo posible; el cliente debe poder desplegarla tanto en servidores dedicados como en entornos compartidos con otra aplicación.

4 DISEÑO

En esta fase se comenta las decisiones tomadas finalmente para la implementación de la aplicación, y se detalla la arquitectura del sistema y cómo interactúan entre sí los diferentes subsistemas que lo componen.

El lenguaje de programación utilizado es PHP, con el framework Laravel. Una de las ventajas principales de usar PHP frente a Python es la experiencia previa en Ezzing Solar con el lenguaje, y las librerías internas existentes con ese lenguaje.

La plataforma se ha desarrollado teniendo en mente la capacidad de mitigar problemas de rendimiento y ofreciendo alta disponibilidad, a través de la replicación de sus componentes.

Para ello, el proveedor de infraestructura elegido es finalmente Amazon Web Services; se desarrollará la aplicación centralizando el estado siempre en: RDS (base de datos MySQL), S3 (imágenes y documentos de usuario) y Redis (como caché de aplicación). La aplicación se desplegará en varios servidores EC2 con Ubuntu, que respondan a las peticiones de los usuarios. Se puede ver un esquema de la arquitectura completa de la aplicación en la Figura 3.

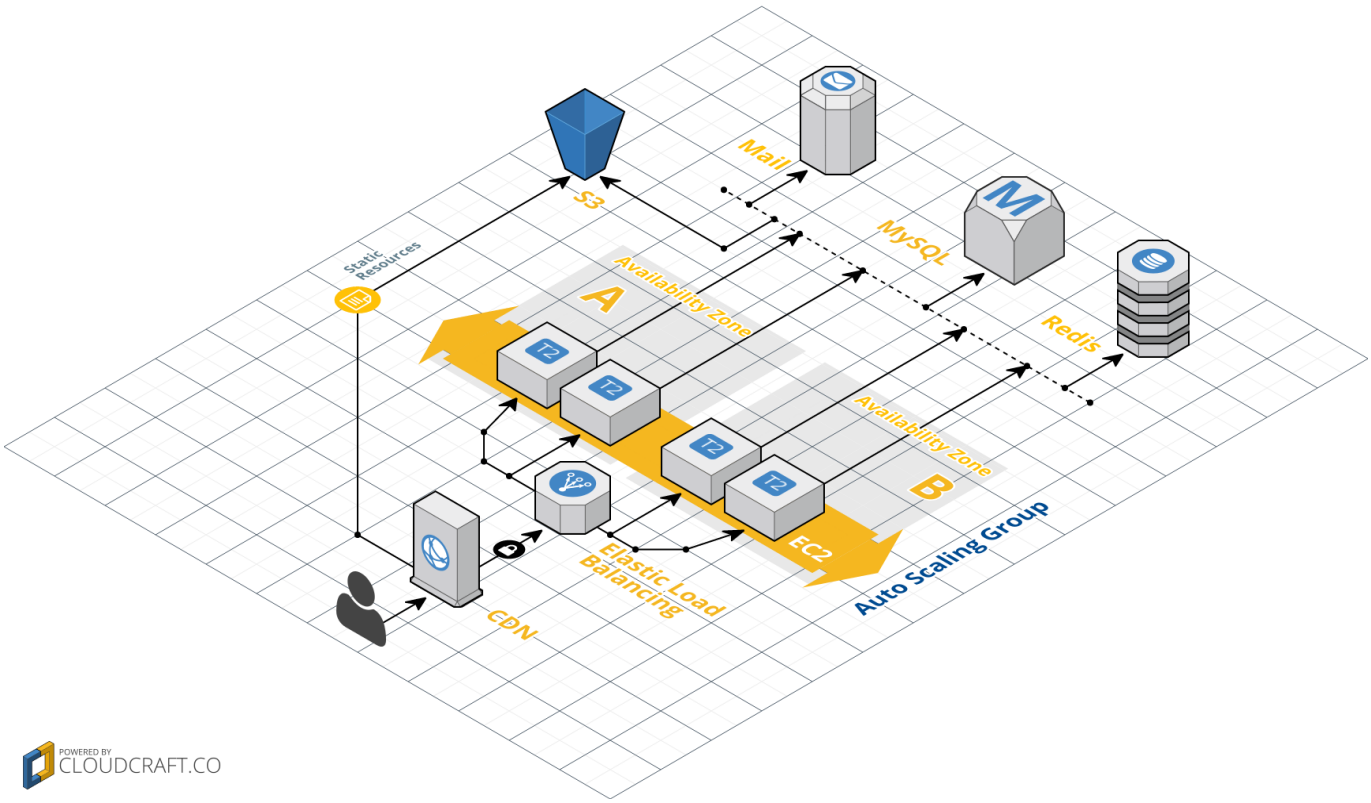


FIGURA 3. ARQUITECTURA DE LA APLICACIÓN

A continuación se detalla los siguientes componentes que forman la plataforma:

4.1 Servidores de la API

El backend de la aplicación se ha desplegado en servidores **EC2** con Ubuntu 14.04, una distribución con mucha utilización en sistemas como el que vamos a construir y para la que se puede encontrar mucha documentación para la instalación de cualquier software. Estos servidores son los que responden a cualquier consulta a la API; con la parte de la plataforma desarrollada en PHP.

Se ha utilizado un **Auto Scaling Group** (ASG) para monitorizar el estado de los servidores, y la utilización de recursos. Cuando los recursos en uso están por debajo o por encima del límite establecido, se cierra algún servidor o se abre alguno nuevo automáticamente.

Además, el ASG se ha configurado para lanzar alertas de salud de los servidores o de utilización de recursos por encima de otro límite establecido, que requieran ver qué ocurre manualmente.

4.2 Datos de usuario

Las imágenes de productos, reportes generados de un proyecto, CSV con información de productos y clientes, etc. son todos datos estáticos de usuario que utilizará la API y es necesario almacenarlos centralizados.

Para conseguir esto se ha utilizado **Amazon S3**; es una solución poco costosa, rápida y con garantías de disponibilidad con la que podremos guardar todo tipo de documentos sin límite de contenido, tanto desde el servidor como desde JavaScript en el navegador a través de su API HTTP. Con esta solución, también podemos replicar los objetos estáticos a través de un CDN (Amazon CloudFront) para mejorar su tiempo de descarga.

4.3 Aplicación web (frontend)

Dado que la interfaz web son solo ficheros estáticos, esto nos permite utilizar también S3 para su distribución, por un coste despreciable y sin tener que mantener un servidor para ello. Para mejorar el tiempo de carga inicial de la aplicación, se ha utilizado CloudFront como CDN para su distribución.

4.4 Base de datos

La base de datos elegida finalmente es MySQL. Se gestiona a través de **Amazon RDS**, para poder disponer de replicación de forma transparente. Esto nos proporciona un mayor rendimiento (habrá copias de solo lectura que quitarán carga de trabajo a la base de datos maestra) y alta disponibilidad en la base de datos.

Este sistema es una base de datos MySQL normal, con lo que no hará falta ningún cambio si en el futuro se quiere migrar a otra MySQL fuera de Amazon.

4.5 SERVIDOR DE CACHEO

Amazon también ofrece abstracciones sobre la base de datos no relacional Redis, pero no están tan probados ni ofrecen tantas ventajas como RDS. Por ese motivo, en este caso se ha abierto un servidor Ubuntu 14.04 normal con Redis configurado manualmente..

5 DESARROLLO

A continuación se detalla los patrones de diseño que se ha seguido durante la implementación, la estructura en que se ha organizado el proyecto y los diferentes módulos que lo componen.

5.1 DESARROLLO ORIENTADO A PRUEBAS

Durante el desarrollo se ha seguido un modelo de desarrollo orientado a pruebas (TDD, **Test Driven Development**), utilizando el framework de automatización de tests PHPUnit [25].

Con este modelo de trabajo, se realizan ciclos cortos de desarrollo en los que se comienza transformando requisitos funcionales en casos de prueba exhaustivos, para a continuación implementar la funcionalidad necesaria para pasar esas pruebas. Cada ciclo de desarrollo termina cuando se pasan todas las pruebas, y se vuelve a empezar eligiendo otros requisitos que transformar en pruebas.

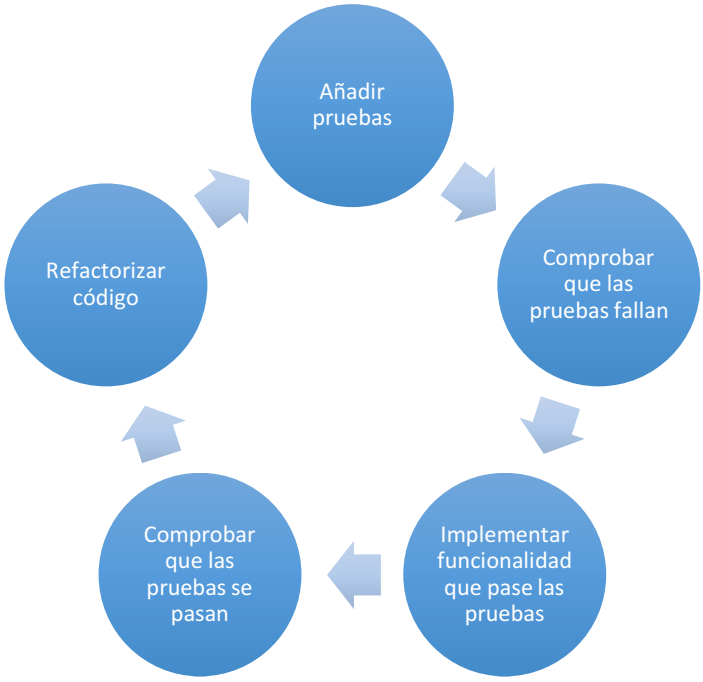


FIGURA 4. CICLO DE DESARROLLO ORIENTADO A PRUEBAS

5.2 DESARROLLO DEL BACKEND

Laravel impone ciertas restricciones a las aplicaciones que lo utilizan, obligando a los desarrolles a ordenar el código siguiendo varios patrones de diseño y criterios. Se puede ver la estructura general de carpetas del proyecto en la Figura 5.

```
├─ app
│  ├─ Events
│  ├─ Http
│  │  ├─ Controllers
│  │  ├─ Kernel.php
│  │  ├─ Middleware
│  │  └─ routes.php
│  ├─ Listeners
│  ├─ Models
│  ├─ Policies
│  ├─ Providers
│  └─ Services
├─ composer.json
├─ composer.lock
├─ config
├─ database
│  ├─ migrations
│  └─ seeds
├─ phpspec.yml
├─ phpunit.xml
├─ public
│  ├─ favicon.ico
│  ├─ index.php
│  └─ robots.txt
├─ resources
│  ├─ lang
│  └─ views
├─ tests
├─ vagrant
└─ Vagrantfile
```

FIGURA 5. ESTRUCTURA GENERAL DEL PROYECTO (BACKEND)

Lo primero que se puede ver es que se organiza el código siguiendo el patrón de diseño **MVC** (Model-View-Controller), separando el código en las siguientes secciones principales:

- **Modelos**. Son tanto las entidades como la lógica de la aplicación.
  - *app/Models/*. Son las entidades que representan elementos en la base de datos.
  - *app/Services/*. Lógica de la aplicación.
  - *app/Policies/*. Lógica de autorización de acciones a entidades para un usuario.
- **Vistas**. Como en nuestra aplicación sólo estamos construyendo una API, la vista solo formatea la información como JSON. En casos diferentes, se encontraría las plantillas de HTML en *resources/views/*.
- **Controladores**. Engloban la definición de las rutas existentes en la API y asociarlas con diferentes clases que llaman a la lógica de la aplicación, y la devuelven a través de las vistas. Se encuentran en:
  - *app/Http/Middleware/*. Definición de acciones genéricas que realizar durante el transcurso de una petición. Pueden ser acciones comunes a ejecutar en todas las peticiones, como inicialización y carga de la sesión del usuario, autenticación, o transformación de errores y excepciones en respuestas de error de HTTP, o acciones especiales para algunas consultas específicas.
  - *app/Http/routes.php*. Definición de rutas existentes, y middleware especiales que ejecutar para algunas de ellas.
  - *app/Http/Controllers/*. Código a ejecutar para cada ruta, desde aquí se llama a los servicios necesarios.

- *app/Http/Kernel.php.* Instalación de los middleware que ejecutar para todas las peticiones.

Laravel también emplea el patrón de **Inyección de Dependencias**, a través de un **Service Container** [26], un contenedor centralizado que se encarga de gestionar la creación de instancias de otras clases. Este contenedor se puede configurar y personalizar cómo se instancia clases específicas.

La existencia del Service Container permite una organización muy limpia del código, y facilita mucho la implementación de pruebas, en las que se puede probar solo partes relevantes de código cambiando dependencias que utilicen por objetos simulados (*mock objects*). Los servicios a instalar en el Service Container se configuran en la carpeta *app/Providers/*.

En la **Error! Reference source not found.** se puede ver los módulos principales del sistema y su relación entre ellos.

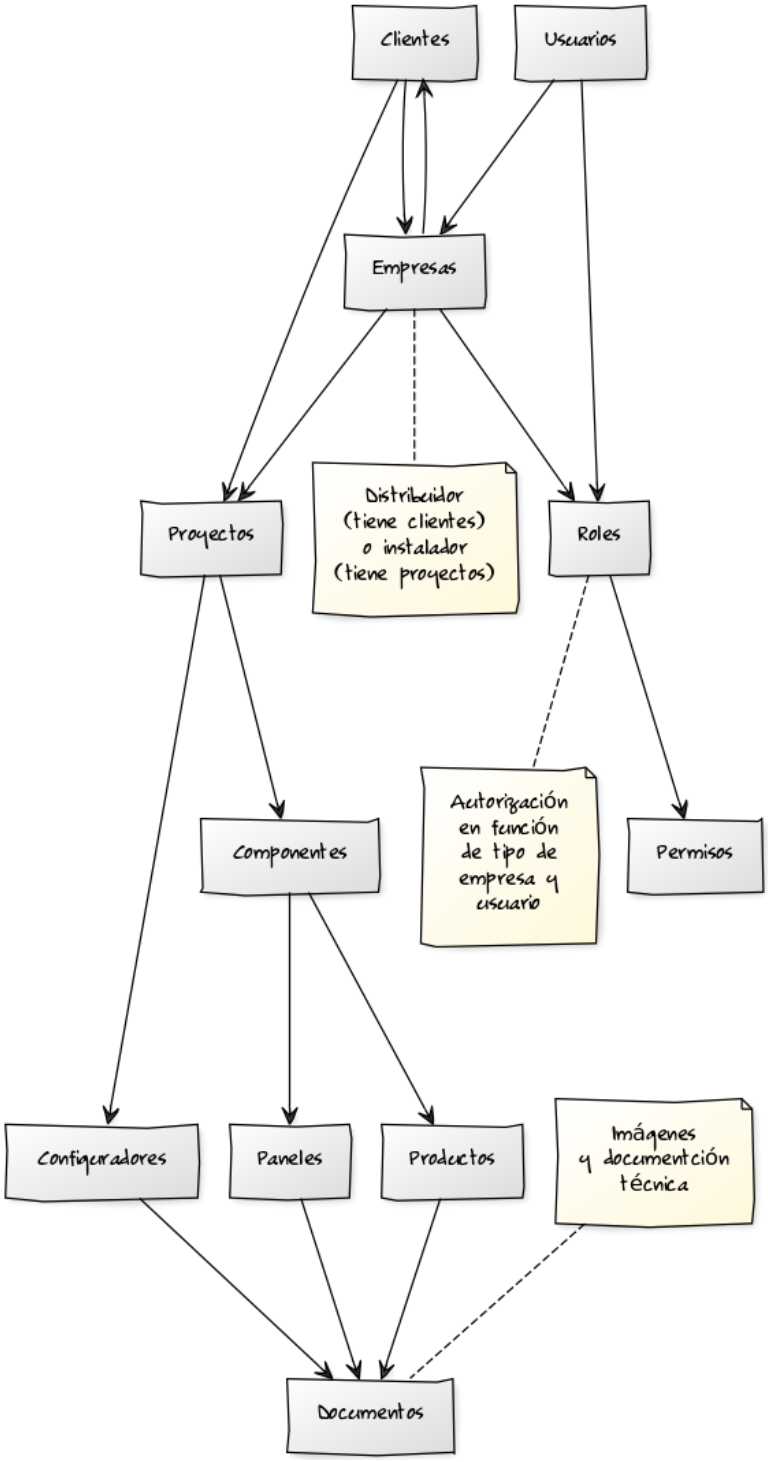


FIGURA 6. MÓDULOS PRINCIPALES DEL SISTEMA

5.3 DESARROLLO DEL FRONTEND

El frontend de la aplicación se ha realizado con Angular. La estructura es más abierta, pero también se utiliza un patrón MVC. La estructura general de ficheros que se ha seguido en el proyecto es la siguiente:

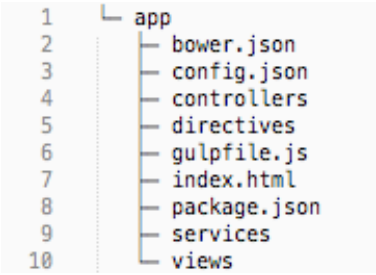


FIGURA 7. ESTRUCTURA GENERAL DEL PROYECTO (FRONTEND)

En este caso, los servicios (*app/services/*) son los únicos modelos que se han utilizado; estos módulos contienen la lógica de aplicación necesaria y las llamadas a la API, y se accederá a los datos devueltos por la API a través de ellos.

Los controladores (*app/controllers/*) definen las rutas existentes en la aplicación de Angular, y se encargan de asociar inputs de la vista (inputs de texto, clicks en botones, etc.) con los servicios apropiados, y de exponer los datos necesarios de un servicio a cada vista.

En Angular, las vistas se encuentran en:

- *app/views/*. Contiene plantillas HTML con expresiones de Angular para dar dinamismo a la página y mostrar información del modelo.
- *app/directives/*. Directivas que utilizar desde las plantillas. Son componentes reutilizables que pueden contar con su propia plantilla HTML, controlador y servicios.

6 PRUEBAS Y RESULTADOS

Durante el desarrollo se ha seguido un desarrollo orientado a pruebas (TDD, **Test Driven Development**) durante el cual se ha realizado extensivas pruebas. Seguir esta metodología facilita definir los límites de cada ciclo de desarrollo al principio de éste, así como el futuro mantenimiento del código.

Estas pruebas son una mezcla de pruebas automatizados (test unitarios y de aceptación) y manuales (pruebas de aceptación).

6.1 PRUEBAS UNITARIAS

Las pruebas unitarias son las pruebas de más bajo nivel, y su función es comprobar que el código de la aplicación se comporta correctamente dados los diferentes tipos de datos de entrada esperados, y que response correctamente a errores que se prevé que puedan ocurrir.

Algunas pruebas representativas que se han realizado satisfactoriamente son:

- Comprobar la lógica de autenticación con usuarios válidos e inválidos.
- Creación y edición de proyectos, devolviendo errores adecuados de validación al intentar repetir un campo único o darle un valor inválido a un campo.
- Subida atómica de fichas técnicas de un producto, de forma que el documento sea accesible desde todos los servidores en cuanto se guarda en la base de datos la referencia a su localización.
- Creación de clientes instaladores bajo una empresa distribuidora, y visibilidad en la distribuidora de los proyectos que crean los instaladores.

6.2 PRUEBAS DE INTEGRACIÓN

Las pruebas de integración se encargan de comprobar que los diferentes módulos de una aplicación que ya han pasado pruebas unitarias interactúan correctamente entre sí para llevar a cabo una funcionalidad de más alto nivel.

Algunas pruebas de integración representativas realizadas son:

- Al obtener un listado de proyectos a través de la API un usuario solo ve aquéllos para los que tiene permisos de lectura.

- Al editar proyectos a través de la API un usuario solo puede editar aquéllos para los que tiene permisos de escritura.

- Consultas a la API sin autenticar o sin permisos suficientes devuelven códigos adecuados de error.

6.3 Pruebas de validación

Mientras que las pruebas unitarias y de integración se encargan de comprobar la calidad del código, las pruebas de validación determinan si la implementación realizada cumple su propósito de cara a los usuarios.

Las pruebas que se han realizado en esta fase son las siguientes:

- Un usuario administrador puede autenticarse y cerrar sesión.
- Un usuario administrador puede invitar nuevos administradores y crear nuevos usuarios clientes, agrupándolos por compañía a la que pertenecen.
- Un usuario administrador puede bloquear usuarios clientes existentes en la plataforma.
- Un usuario administrador tiene visibilidad y puede editar los proyectos realizados por clientes.
- Un usuario administrador puede crear nuevos productos manualmente, importar productos en bloque a través de un CSV y editar los productos existentes.
- Un usuario cliente puede crear nuevos proyectos.
- Un usuario cliente puede editar proyectos que le pertenecen.

6.4 Pruebas de aceptación

Las pruebas de aceptación, al igual que las de validación, determinan si el producto desarrollado es el adecuado. Su finalidad es comprobar que el producto satisface las especificaciones y requisitos definidos.

En este proyecto se han realizado pruebas de aceptación a través de los clientes finales. Ha habido feedback positivo hasta el punto de que posteriormente se han realizado otros proyectos relacionados con ellos.

7 CONCLUSIONES Y TRABAJO FUTURO

7.1 CONCLUSIONES

La realización de este proyecto es muy beneficiosa para Ezzing Solar, puesto que permite partir de una base ya construida para cada nuevo proyecto de este estilo. Es una plataforma que se puede seguir extendiendo con el tiempo, y las tecnologías usadas permiten que cada cambio se pueda también actualizar en las aplicaciones ya existentes, como parte de su mantenimiento.

Por el tipo de desarrollo que implica, no hay ninguna limitación a la hora de continuar desarrollándolo. Siempre que continúe utilizándose en proyectos merecerá la pena invertir más tiempo y recursos en desarrollar una base más compleja que vaya abaratando cada vez más los futuros desarrollos.

Este proyecto ha resultado también muy interesante a nivel personal, puesto que ha dado la oportunidad de investigar un gran abanico de servicios muy utilizados en el desarrollo web, y es una gran oportunidad para evaluarlos y probarlos.

7.2 TRABAJO FUTURO

Debido a que esta aplicación tiene el objetivo de poder ser reutilizable en diferentes proyectos, no es bueno que exista una dependencia con Amazon Web Services. Una mejora beneficiosa sería añadir una capa de abstracción con los servicios de AWS que actualmente se están utilizando directamente, como la gestión de documentos con S3. Añadiendo una capa de abstracción que muestre una interfaz de un sistema de ficheros, ocultando si se utiliza S3 u otra solución, facilitaría cambiar rápidamente este servicio por otro para un proyecto que fuera a desplegarse en otra plataforma.

Sería beneficioso para la aplicación incorporar un sistema flexible de indexado de información, para que un usuario pudiera filtrar un listado de clientes, proyectos o productos más fácilmente. Para realizar esto, sería interesante investigar servicios como ElasticSearch, o Solr, y ver qué alternativas existen.

Otra mejora para reducir la complejidad de la aplicación y hacerla más extensible y depurable sería realizar la comunicación entre servicios a través de eventos, por medio de una cola de trabajo (servicios como RabbitMQ, ActiveMQ, Beanstalkd, etc.). Hacerlo así facilitaría la comunicación a través de eventos; a replicar, añadir o quitar servicios; y a monitorizar el estado y el rendimiento de cada parte que compone la plataforma.

8 REFERENCIAS

[1] Oracle Corporation. MySQL. [Online]. https://www.mysql.com/

[2] PostgreSQL. PostgreSQL: The world's most advanced open source database. [Online]. https://www.postgresql.org/

[3] The MariaDB Foundation. MariaDB.org - Ensuring continuity and open collaboration. [Online]. https://mariadb.org/

[4] CloudFlare. CloudFlare - The web performance & security company. [Online]. https://www.cloudflare.com/

[5] Amazon Web Services. Amazon CloudFront – Content Delivery Network (CDN). [Online]. https://aws.amazon.com/cloudfront/

[6] Thumbor. Thumbor - Open-source photo thumbnail service by globo.com. [Online]. https://github.com/thumbor/thumbor

[7] Varnish HTTP Cache. Varnish HTTP Cache. [Online]. https://varnish-cache.org/

[8] NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy. [Online]. https://www.nginx.com/

[9] Memcached. Memcached - a distributed memory object caching system. [Online]. https://memcached.org/

[10] Redis. Redis. [Online]. http://redis.io/

[11] PyPA. Pip: Python Package Index. [Online]. https://pypi.python.org/pypi/pip

[12] Django Software Foundation. The Web framework for perfectionists with deadlines | Django. [Online]. https://www.djangoproject.com/

[13] Armin Ronacher. Flask (A Python Microframework). [Online]. http://flask.pocoo.org/

[14] Composer. Composer - Dependency Management for PHP. [Online]. https://getcomposer.org/

[15] SensioLabs. Symfony, High Performance PHP Framework for Web Development. [Online]. https://symfony.com/

[16] Qiang Xue. Yii PHP Framework: Best for Web 2.0 Development. [Online]. http://www.yiiframework.com/

[17] Taylor Otwell. Laravel - The PHP Framework For Web Artisans. [Online]. https://laravel.com/

[18] AngularJS. AngularJS — Superheroic JavaScript MVW Framework. [Online]. https://angularjs.org/

[19] BackboneJS. Backbone.js. [Online]. http://backbonejs.org/

[20] Facebook. A JavaScript library for building user interfaces | React. [Online]. https://facebook.github.io/react/

[21] JetBrains. PhpStorm IDE. [Online]. https://www.jetbrains.com/phpstorm/

[22] JetBrains. PyCharm IDE. [Online]. https://www.jetbrains.com/pycharm/

[23] HashiCorp. Vagrant by HashiCorp. [Online]. https://www.vagrantup.com/

[24] Amazon Web Services (AWS) - Cloud Computing Services. [Online]. https://aws.amazon.com/

[25] PHPUnit, The PHP Testing Framework. [Online]. https://phpunit.de/

[26] Taylor Otwell. Service Container - Laravel - The PHP Framework For Web Artisans. [Online]. https://laravel.com/docs/5.1/container

9 ANEXOS

A. ESTRUCTURA COMPLETA DEL BACKEND

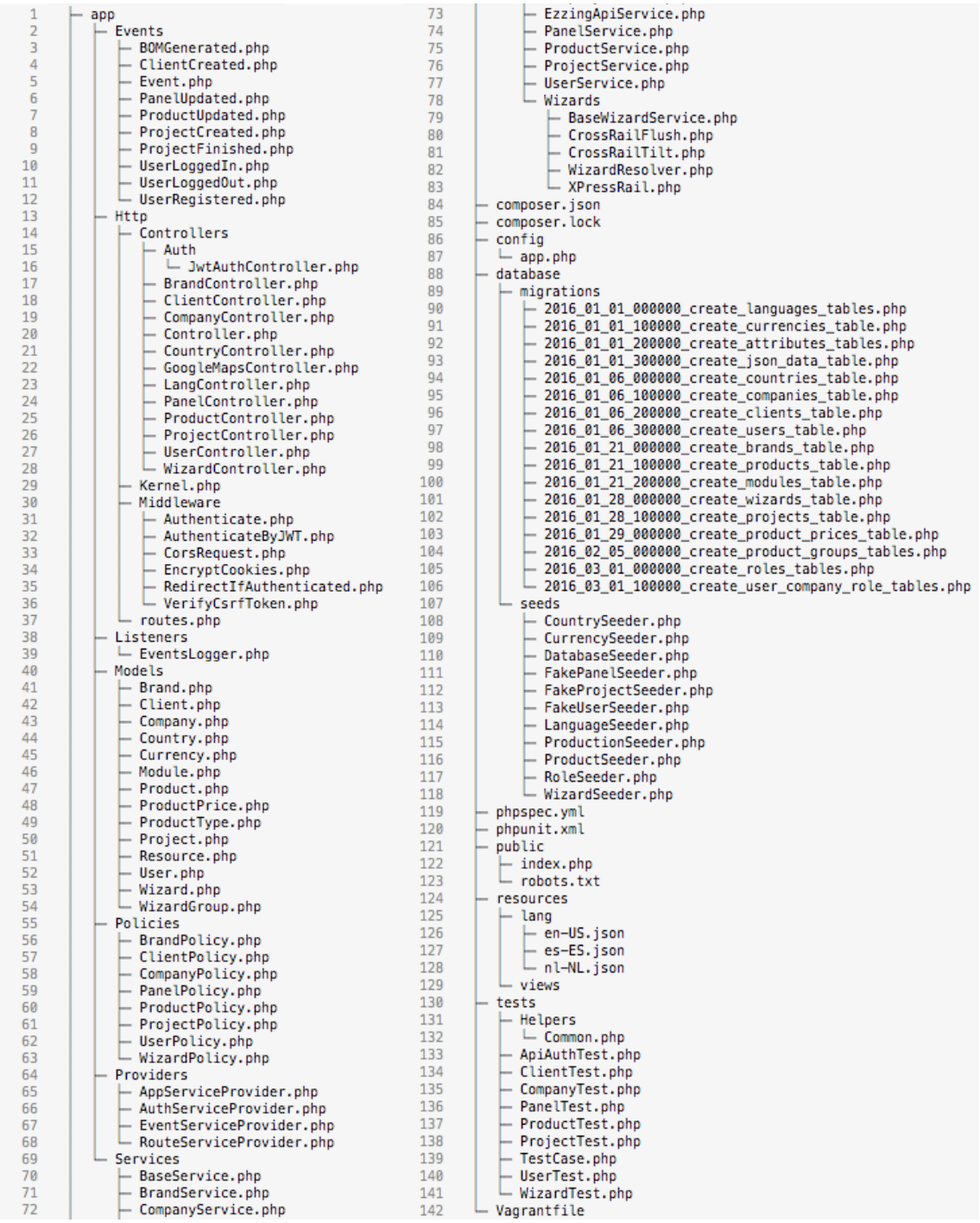


FIGURA 8. DETALLE ESTRUCTURA DEL PROYECTO (BACKEND)

B. ESTRUCTURA COMPLETA DEL FRONTEND

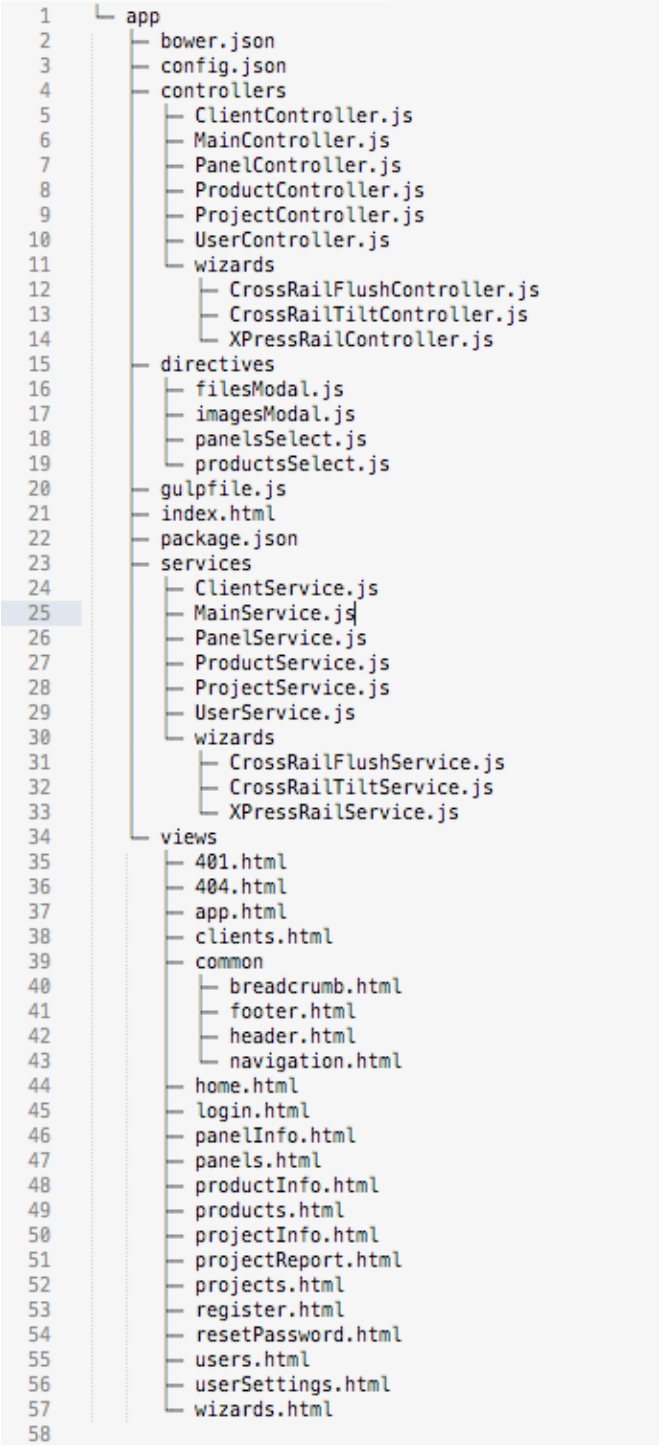


FIGURA 9. DETALLE ESTRUCTURA DEL PROYECTO (FRONTEND)

C. MANUAL DE USUARIO DE LA HERRAMIENTA

Acceso a la herramienta

Introduce tu nombre de usuario y contraseña. Si no tienes, sigue el link de 'Forgot password' para registrarte.

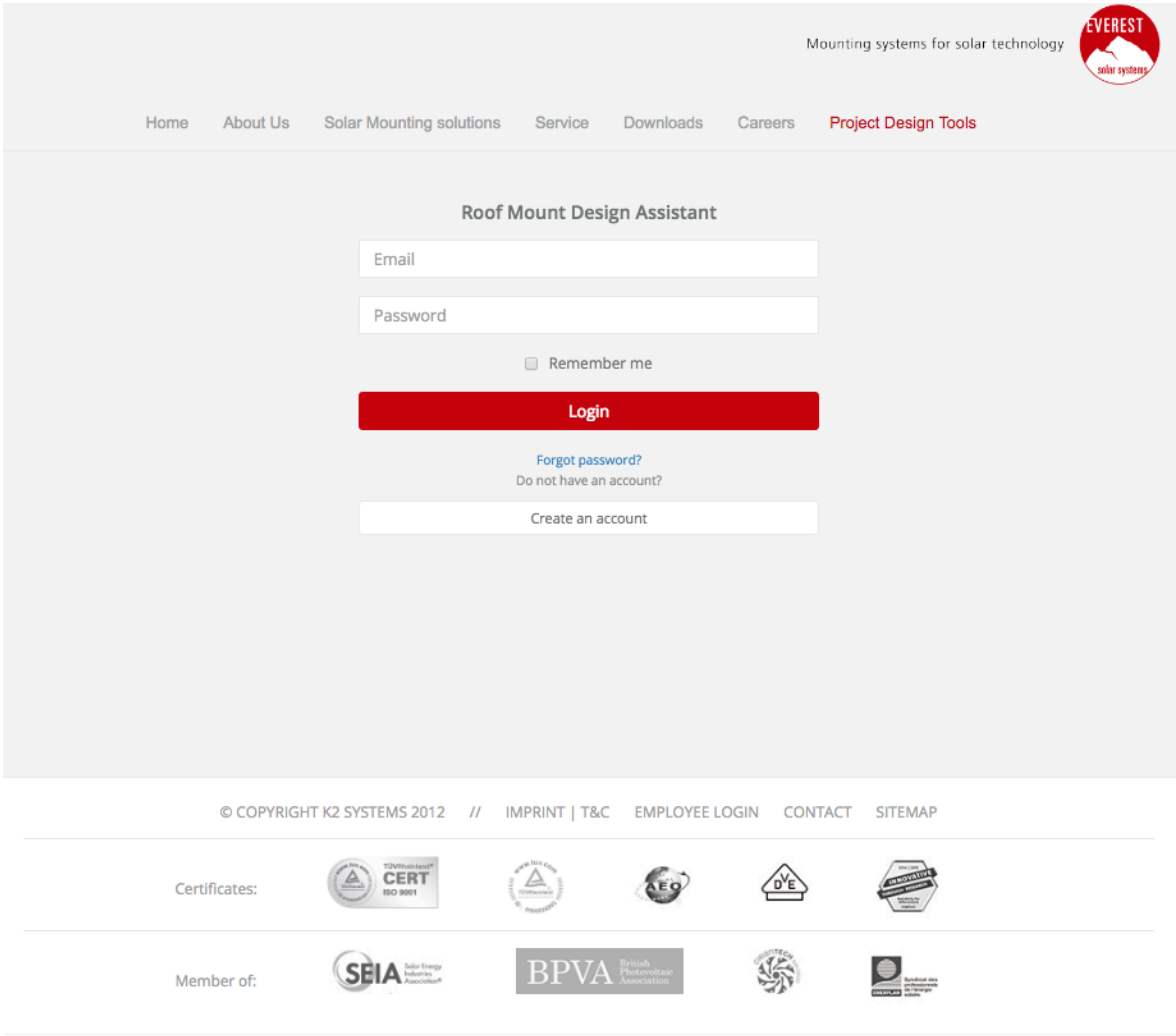


FIGURA 10. ACCESO AL SISTEMA

Usuario distribuidor

Si eres un usuario perteneciente a una empresa distribuidora, en tu pantalla de inicio verás información de las cuentas de instaladores asociados a tu empresa (Figura 11).

A través del menú de la izquierda, podrás acceder al listado de administración de clientes y usuarios de tu cuenta y las cuentas instaladoras asociadas (Figura 13).

También podrás acceder al listado de administración de proyectos (Figura 12).

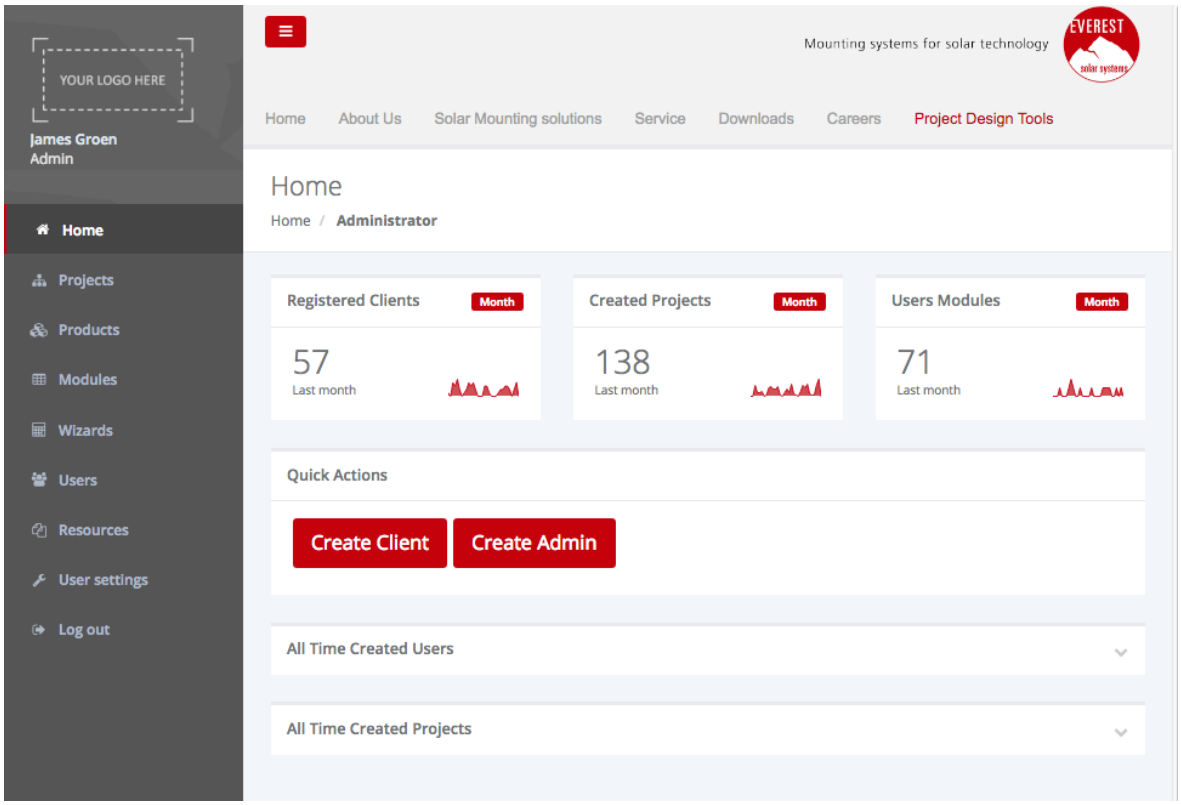


FIGURA 11. VENTANA DE INICIO DISTRIBUIDOR

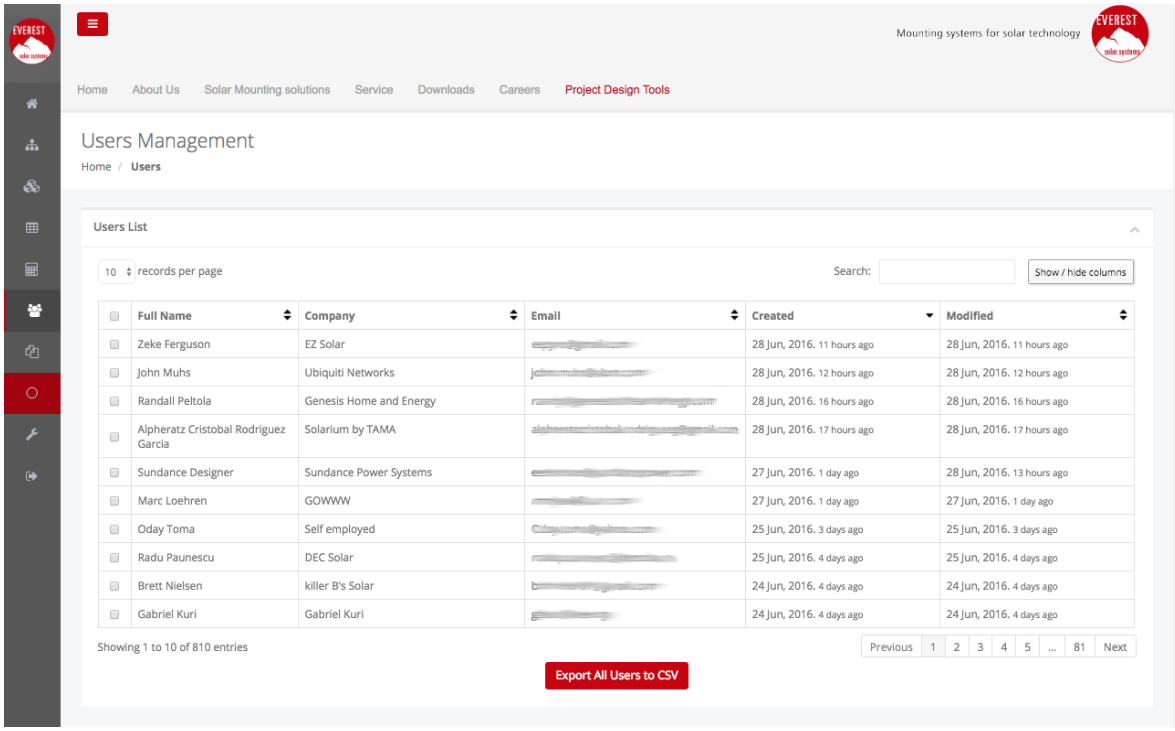


FIGURA 13. ADMINISTRACIÓN DE USUARIOS

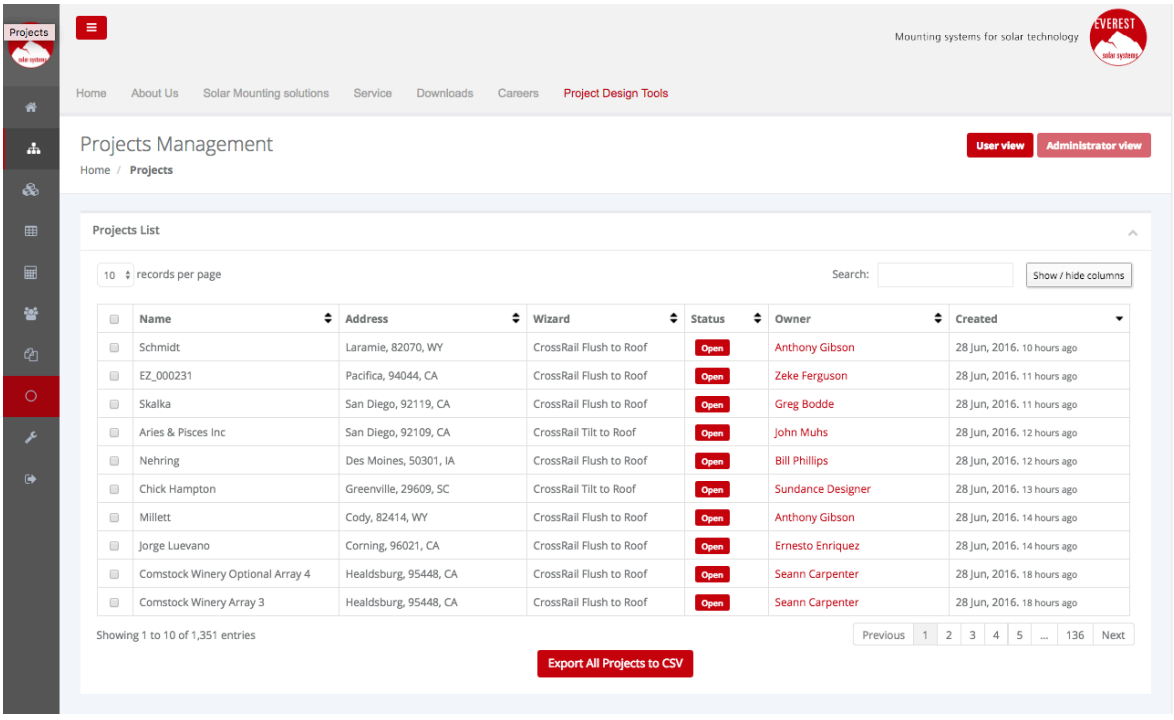


FIGURA 12. ADMINISTRACIÓN DE PROYECTOS

Usuario instalador

Si eres un usuario perteneciente a una empresa instaladora, en tu pantalla de inicio verás accesos directos a la creación de proyectos (Figura 14).

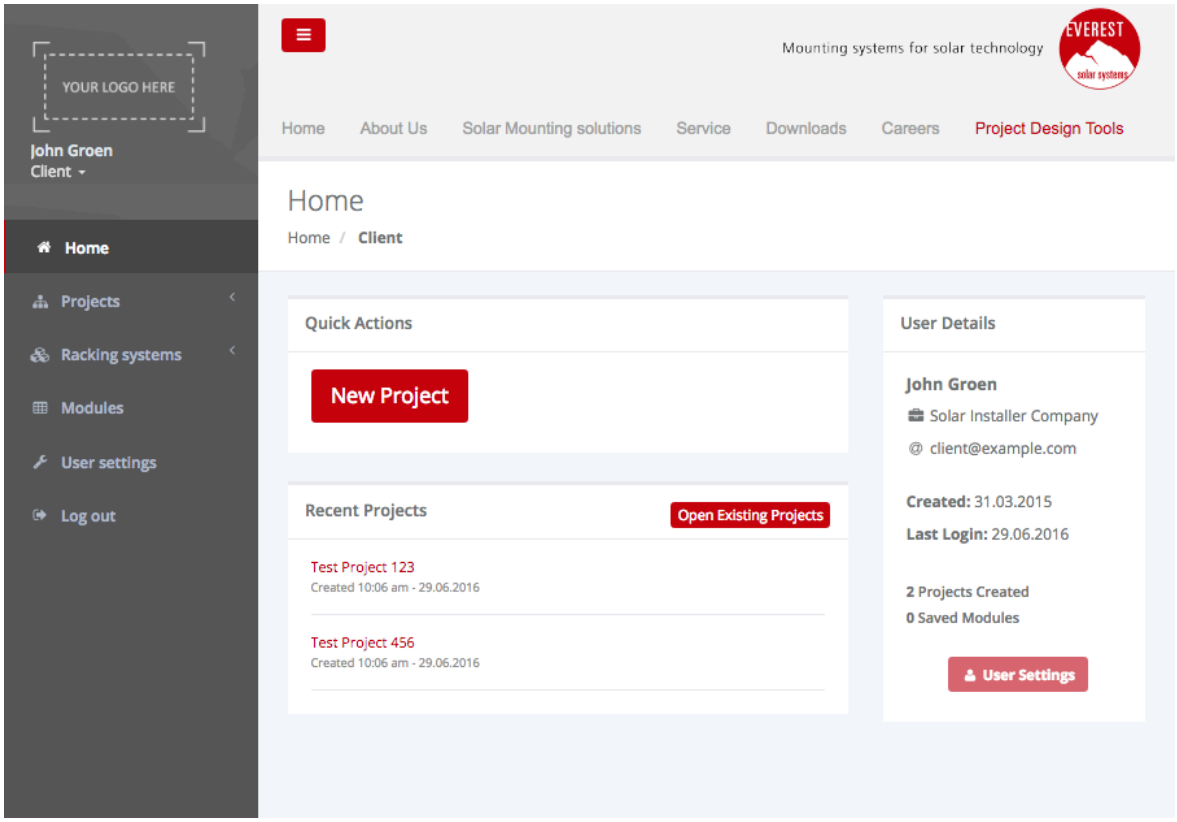


FIGURA 14. VENTANA DE INICIO INSTALADOR

Desde ahí podrás listar y editar tus proyectos anteriores (Figura 15), y acceder a los productos disponibles (Figura 16).

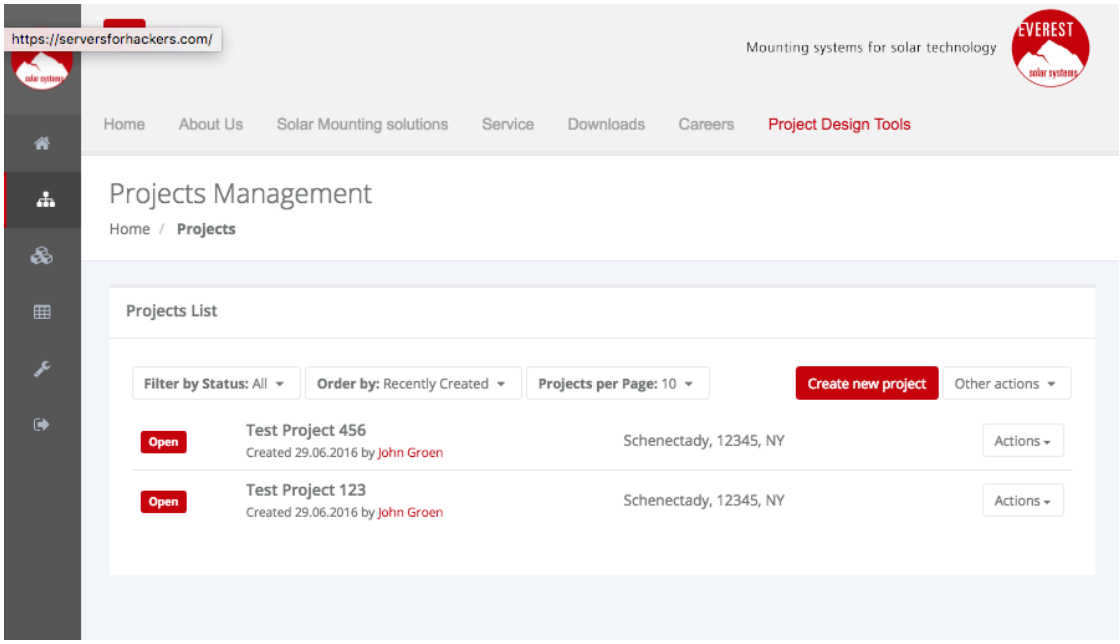


FIGURA 15. LISTADO PROYECTOS DE UN INSTALADOR

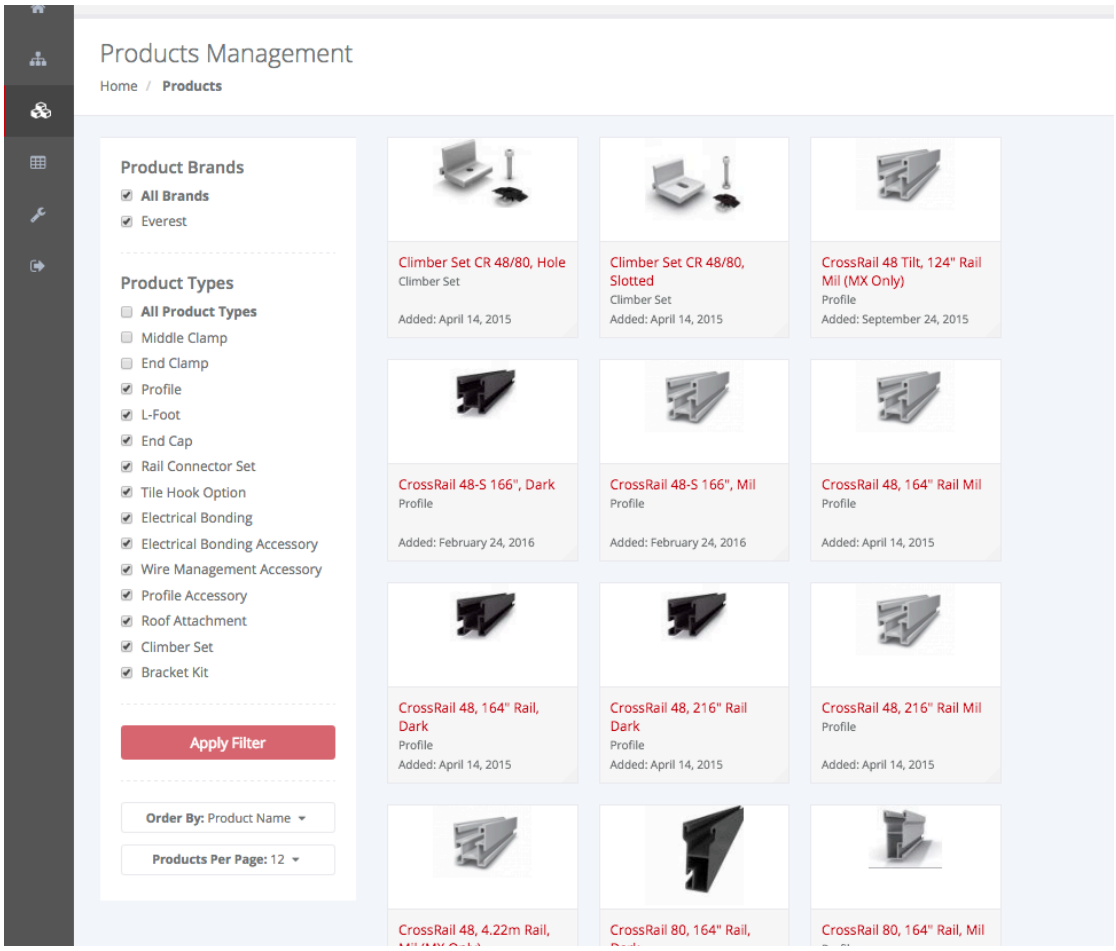


FIGURA 16. LISTADO DE PRODUCTOS

Por último, un usuario instalador podrá acceder desde la pantalla de inicio o el menú de la izquierda al configurador de creación de proyectos. Esto abrirá una aplicación interactiva en el navegador, en la que se preguntará al usuario el tipo de configurador que ejecutar, y diferentes parámetros relativos a éste.

Primero, se pedirá información del área en el que ejecutar el configurador (Figura 17), y se elegirá el tipo de configurador que ejecutar (*Mounting Type*).

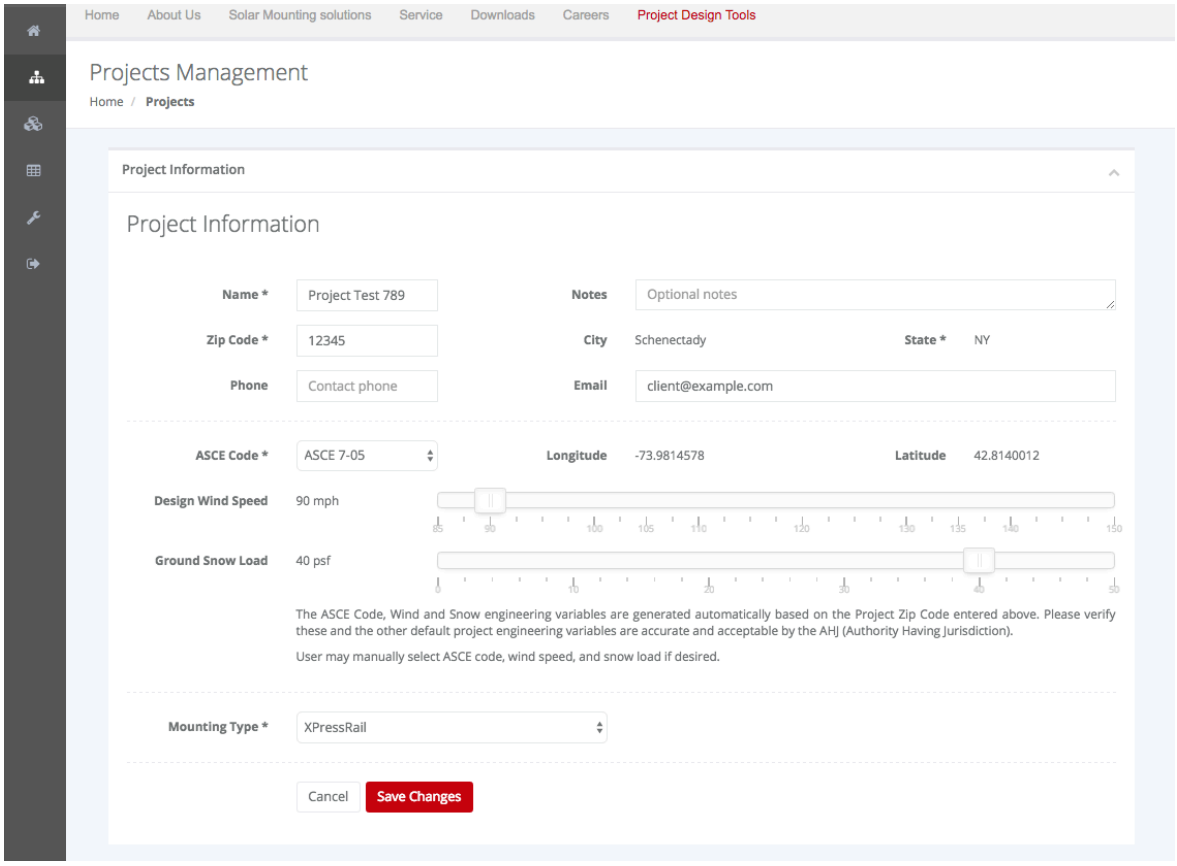


FIGURA 17. INFORMACIÓN DEL PROYECTO Y TIPO DE CONFIGURADOR

Una vez cargado el tipo de configurador, se pedirá al usuario información sobre el edificio en el que irá la instalación.

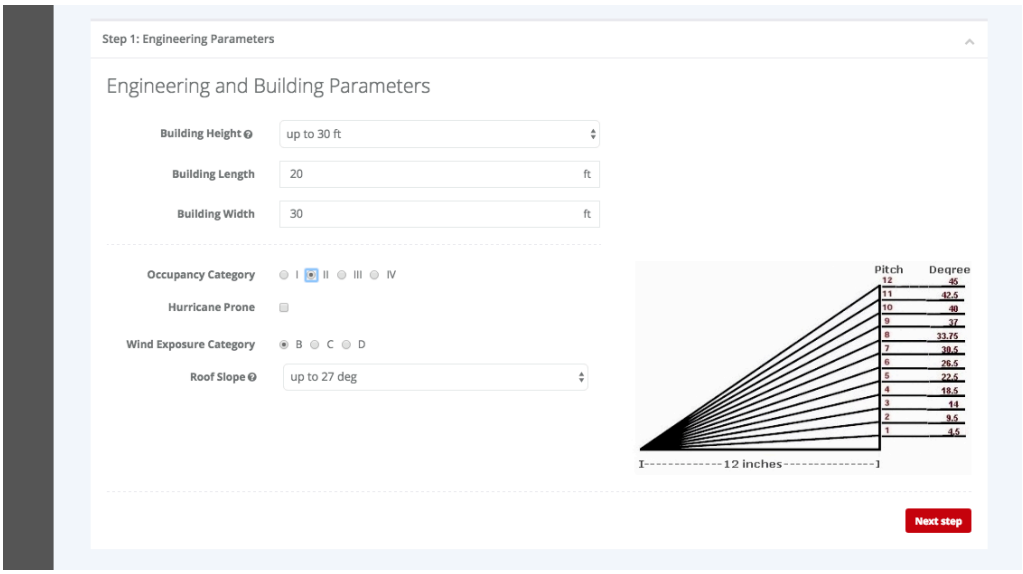


FIGURA 18. PARÁMETROS DEL EDIFICIO

A continuación se elige el tipo de panel que utilizar, y se muestra cálculos de rendimiento de los componentes según los parámetros que ha elegido el instalador, para darle la opción de hacer cambios.

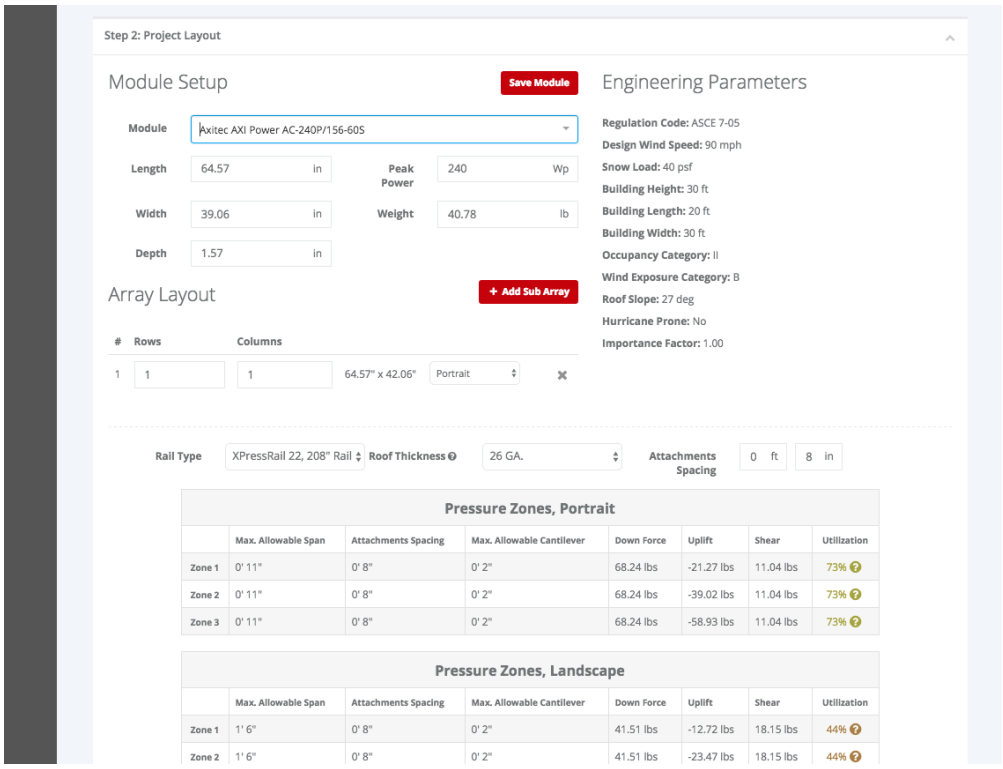


FIGURA 19. TIPO DE MÓDULO Y DIMENSIONES DE LA INSTALACIÓN

Por último, se muestra el listado de productos necesarios para realizar la instalación. El usuario podrá cambiar los accesorios que quiera que se añadan e incluso editar manualmente la cantidad de productos listados.

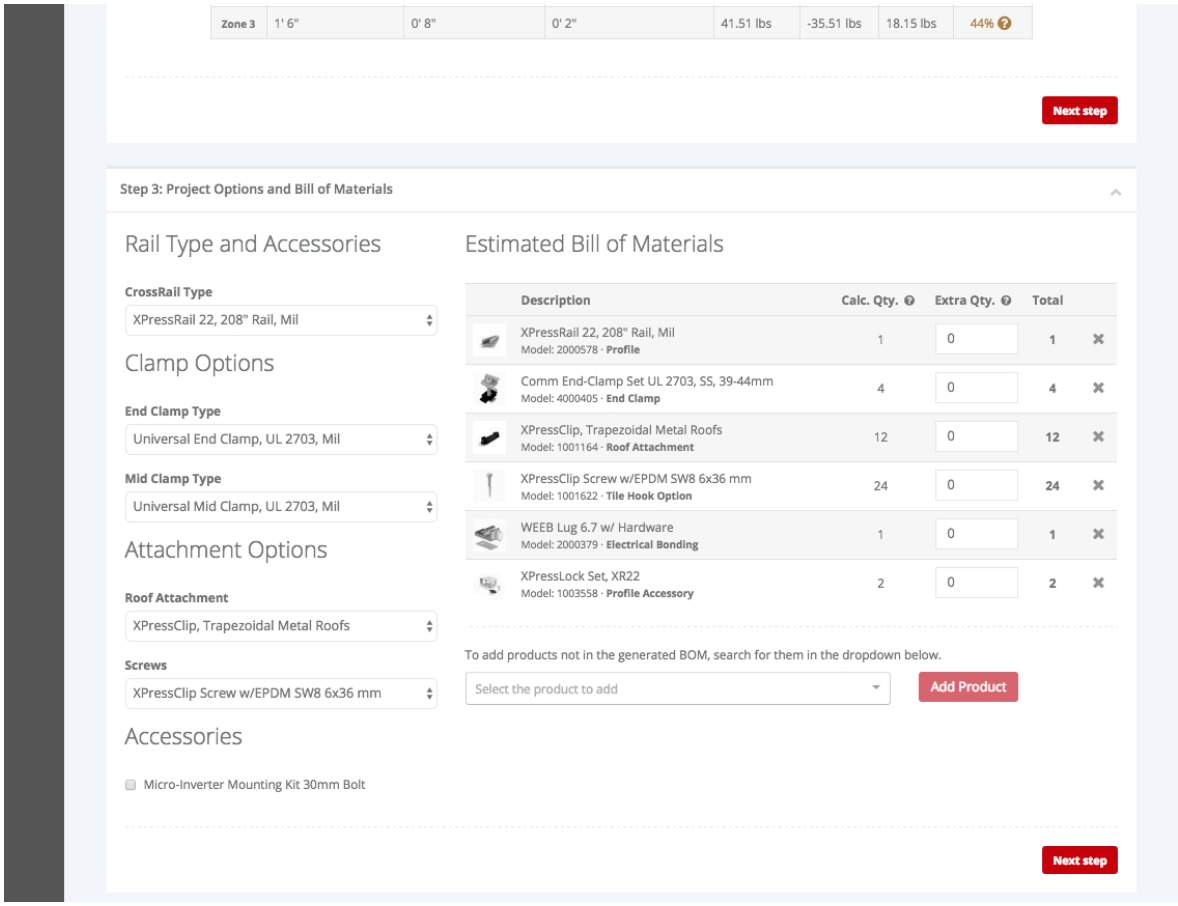


FIGURA 20. OPCIONES DEL LISTADO DE COMPONENTES GENERADO

Finalmente, el usuario podrá descargar un reporte con el listado de productos necesarios, o guardar el proyecto para volver más adelante.

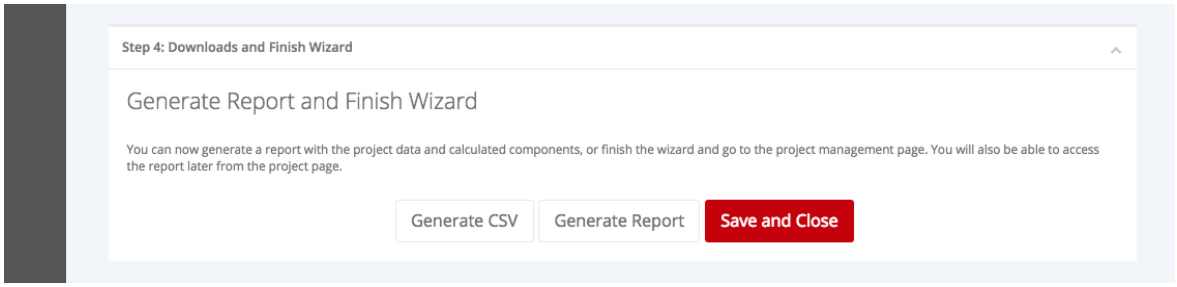


FIGURA 21. EXPORTAR EL PROYECTO O SALIR