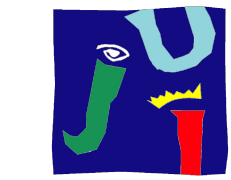

Grado en Ingeniería Informática

Trabajo Final de Grado

Desarrollo de 480 CRM en Symfony 3

Autor:
Miguel Amela Llobet

Supervisor:
Sergio Aguado González
Tutor académico:
Reyes Grangel Seguer

Fecha de lectura: 23 de marzo de 2018
Curso académico 2017/2018

Resumen

Este documento muestra el Trabajo de Fin de Grado realizado durante la estancia en prácticas del Grado en Ingeniería Informática de la Universitat Jaume I. La estancia se ha llevado a cabo por el autor de la memoria en la empresa Soluciones Cuatroochenta S.L. El proyecto desarrollado consiste en una aplicación web de CRM que será usada por la propia empresa.

Este CRM sirve como herramienta para realizar un seguimiento de las oportunidades de negocio de la empresa. De esta forma, se mejora este proceso y permite obtener más información para que estas oportunidades se conviertan en un proyecto a desarrollar por la empresa.

Esta aplicación web se ha implementado con el framework Symfony, concretamente en la versión 3.3. Este framework está desarrollado en PHP y ofrece velocidad y flexibilidad debido a su arquitectura, que está completamente desacoplada.

En este documento se exponen los aspectos más importantes del proceso de planificación, análisis e implementación con respecto al desarrollo de esta aplicación web, que se ha llevado a cabo para satisfacer los requisitos de la empresa.

Palabras clave

Symfony 3, CRM, desarrollo web, PHP, 480 CRM

Keywords

Symfony 3, CRM, web development, PHP, 480 CRM

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

En la estancia en prácticas de la asignatura EI1054 del Grado en Ingeniería Informática de la UJI se ha de llevar a cabo un proyecto en el que se muestren los contenidos formativos recibidos y las competencias adquiridas durante los estudios de este Grado. En este documento se presenta la estancia llevada a capo por Miguel Amela en la empresa Soluciones Cuatroochenta S.L. [30]. Empresa situada en el Espaitec (Parque Científico, Tecnológico y Empresarial de la Universitat Jaume I), concretamente en el edificio Espaitec 2. Cuatroochenta está especializada en el desarrollo integral de aplicaciones para smartphones y tablets y programación avanzada a medida para mejorar procesos de trabajo.

Cuatroochenta nació en el año 2011 y aunque tenga una corta edad, se ha asentado en el mercado del desarrollo de aplicaciones, captando la atención de los expertos en el sector, consiguiendo ser seleccionada para formar parte del Entorno Pre-Mercado [27]. Este entorno sirve para que las startups puedan acceder a nuevos inversores y conozcan el funcionamiento del mercado bursátil, potenciando el acceso a la inversión privada. De esta forma, podrá crecer aún más de lo que lo ha hecho, ya que cuenta con un total de siete oficinas situadas en España, Panamá, Bogotá, Argentina, Holanda, Estados Unidos e Italia.

Gran parte del éxito de Cuatroochenta se debe a las aplicaciones propias que han desarrollado para la gestión interna de los proyectos que llevan a cabo, para poder tener el control del estado de los proyectos y poder actuar en base a estos. Una de las herramientas que utilizan para este fin, es un CRM (Customer Relationship Management) hecho a medida por la propia empresa. Un CRM es un programa que permite gestionar la información de los clientes de una empresa. Además, ofrece también la posibilidad de gestionar internamente la empresa a través de diferentes módulos como pueden ser proveedores, campañas, oportunidades, facturas, etc. y otros relacionados con procesos de compra y venta [7]. El uso principal que recibe el CRM de Cuatroochenta es el de dar de alta un cliente potencial que pide información y/o presupuesto para un proyecto. Cuando la gestión avanza y es probable vender ese proyecto, se da de alta a este cliente potencial como Cuenta, donde se añade toda la información de la empresa y se crea una Oportunidad vinculada a esa cuenta.

Con el crecimiento empresarial que ha experimentado Cuatroochenta, el CRM que estaba usando ha quedado un poco desfasado y no es posible aprovecharlo más para aumentar el rendimiento de los trabajadores que lo utilizan. Además, Symfony 1, la tecnología con la que está desarrollado el CRM, ha quedado obsoleto. Por lo tanto, existe la necesidad de mejorar este CRM para aumentar el rendimiento de la empresa. De dicha necesidad nace este proyecto, que consiste en el desarrollo de 480 CRM en Symfony 3, adaptándolo para que aumente el rendimiento en esta tecnología.

1.2. Objetivos del proyecto

El principal objetivo del proyecto es el desarrollo del 480 CRM utilizando Symfony 3. Este CRM contará con un portal web desde el que se podrán llevar a cabo todas las funcionalidades de esta aplicación.

Dicho objetivo se puede desglosar en los subobjetivos que lo forman:

- Formativos: adquirir los conocimientos necesarios para poder desarrollar una aplicación web utilizando el framework Symfony 3, y estudiar el Bundle SgDatatablesBundle [20] y el Bundle FOSUserBundle [10]. Estos dos Bundles son utilizados por la empresa en algunas de las aplicaciones que desarrollan y son necesarios para este proyecto.
- Analizar el funcionamiento del CRM que actualmente utilizan en Cuatroochenta para tratar de adaptarlo a Symfony 3.
- Utilizar los diseños de las interfaces proporcionados por un diseñador de Cuatroochenta para integrarlos en el CRM.
- Desarrollar las funcionalidades analizadas del CRM actual en Symfony 3.
- Integrar la aplicación del CRM con la aplicación web de presupuestos, desarrollada en Symfony 3, que sirve para crear presupuestos para los proyectos y, además, crea en Google Drive una presentación en la cuál se puede observar dicho presupuesto.
- Desarrollar la aplicación web en español e inglés.

1.3. Estructura de la memoria

Además de este primer capítulo de introducción, esta memoria se estructura de la siguiente forma. En el capítulo 2, se presenta de forma más detallada el proyecto, juntamente con las tecnologías usadas. En el capítulo 3, se detalla la planificación seguida en el desarrollo del proyecto. En este capítulo también se muestra la metodología empleada, la planificación, estimación de recursos y costes del proyecto y el seguimiento del proyecto que se ha realizado. En el capítulo 4, se presenta el análisis y el diseño del proyecto. En el capítulo 5, se explican los detalles de la implementación y las pruebas realizadas. Por último, en el capítulo 6, se muestran las conclusiones del proyecto.

Capítulo 2

Descripción del proyecto

2.1. CRM

El término CRM se traduce literalmente como la gestión de las relaciones con los clientes. No obstante, va más allá y cambia la manera de realizar el proceso de ventas de una empresa, ya que esta herramienta ofrece muchas ventajas y agiliza este proceso. Por lo tanto, el CRM nos permite gestionar las relaciones comerciales con los clientes, proveedores, comerciales y los encargados del marketing, centrándonos principalmente en los clientes [8].

Para entender cómo funciona el CRM, se van a presentar los actores que aparecen en este software y la forma en qué interactúan.

En primer lugar, como bien indica el nombre de este tipo de software, uno de los actores es el cliente sobre el que se guarda información. Normalmente, en los CRM se los denomina cómo cuentas. Por lo tanto, se puede entender que una cuenta es como una empresa cliente y, cada uno de los trabajadores de la empresa cliente, se los denomina contactos. Los contactos se pueden clasificar en:

- Prospectos: contactos interesados en los productos que ofrece la empresa, que todavía no han sido contactados.
- Clientes potenciales: contactos interesados en los productos que ofrece la empresa y ya ha sido corroborado este interés.
- Clientes activos: clientes que han comprado un producto a la empresa y mantienen relación con la empresa.
- Clientes inactivos: contactos que han comprado algún producto a la empresa, pero han dejado de comprar productos a la misma.

Para aumentar la información respecto a los clientes y poder facilitar la relación con ellos, se tiene las actividades. Algunos ejemplos de actividades son las llamadas, envíos de correos,

reuniones, etc. De esta forma se puede guardar información personal del cliente y poder ofrecerle un trato más personal.

Después de esto, se presenta otro elemento, las campañas promocionales. Con toda la información que se almacena de los clientes, nos permite gestionar las campañas y poderlas destinar a los clientes que les pueda interesar y, de esta forma, lograr efectividad en este tipo de publicidad [26].

Por último, para intentar fidelizar a los clientes, se usa el equipo de soporte al cliente. La labor principal es la de resolver las incidencias que puedan surgir con el cliente consiguiendo su satisfacción.

Una vez vistos todos los actores que interactúan con el CRM se puede ver el CRM como un software que nos permite recoger información de los clientes de una empresa para tratar de satisfacer sus necesidades con los productos o servicios que vende la empresa.

2.2. CRM Cuatroochenta (480 CRM)

El CRM usado por los trabajadores de Cuatroochenta, denominado 480 CRM, está desarrollado en Symfony, concretamente en la versión 1. Symfony es un framework de PHP que se utiliza para desarrollar aplicaciones web y que, debido a su arquitectura completamente desacoplada, permite añadir o eliminar Bundles para generar nuevas funcionalidades en la aplicación. Un Bundle es un directorio jerarquizado que contiene todo tipo de archivos. Se trata de una librería o un plugin que permite añadir nuevas funcionalidades [28].

El CRM se utiliza para realizar un seguimiento de las oportunidades de las cuentas de los clientes, para tratar de consolidar la oportunidad y que se pueda llevar a cabo un proyecto a partir de esta. Una oportunidad es un proyecto de desarrollo de software. La oportunidad, a su vez, se puede clasificar en diferentes tipos de producto. Por ejemplo, desarrollo web. Por otra parte, una oportunidad tiene un estado para conocer su evolución y tratar de anticiparse a las inquietudes del cliente para ganar el proyecto. Además, tiene asociados distintos presupuestos según las necesidades de las cuentas de los clientes. Una vez finalizada una oportunidad, también se asigna una factura.

Respecto a las cuentas de los clientes, una cuenta es una empresa cliente, con los datos de la misma. Una cuenta puede ser partícipe de varias oportunidades. Además puede adquirir directamente productos que ofrece Cuatroochenta, sin la necesidad de que sea a través de una oportunidad. Por ejemplo, la adquisición de un dominio web a través de Cuatroochenta. Las oportunidades y la adquisición de productos son las principales fuentes de ingreso de la empresa.

La forma en la que se contacta con las cuentas es a través de los contactos. Los contactos son las personas de las cuentas de los clientes a los que se dirigen los usuarios del CRM para realizar el seguimiento de las oportunidades o la venta de productos. Cuando se contacta con un uno de ellos, se guarda el tipo de actividad y la actividad que se ha hecho para conocer mejor los contactos y personalizar el trato con estos.

En el CRM se recoge la información de los posibles clientes que contactan con la empresa a través de su página web [29]. De esta forma, a través del CRM, se puede asignar un trabajador de la empresa para que gestione las necesidades del cliente y analizar si pueden derivar en la creación de una cuenta y una oportunidad.

En esta aplicación existen tres tipos de usuario:

- Usuario administrador global.
- Usuario gestor de aplicación.
- Usuario gestor de oficina.

Estos tipos de usuarios son creados por el administrador global de la aplicación. Este tipo de usuarios puede gestionar usuarios y gestionar aspectos de configuración como pueden ser las oficinas que tiene la empresa. Además de estas labores, también puede gestionar oportunidades, cuentas, contactos, contactos web y productos, es decir, va a poder gestionar completamente toda la aplicación de CRM.

Por otro lado, el gestor de la aplicación puede gestionar las oportunidades, cuentas, contactos, contactos web y productos. No obstante, no puede gestionar los usuarios ni los aspectos de configuración.

Por último, el gestor de oficina puede gestionar únicamente oportunidades, cuentas, contactos, contactos web y productos que pertenezcan a la oficina en la que trabaja este tipo de usuario. Por ejemplo, un usuario que trabaja en la oficina de España y pertenece a este tipo de usuarios, puede gestionar oportunidades, cuentas, contactos, contactos web y productos de empresas que se han dirigido a la oficina de España para interesarse por alguno de estos elementos.

2.3. Tecnologías

En este apartado se van a detallar las tecnologías que se han utilizado para el desarrollo de 480 CRM y algunas herramientas utilizadas durante el desarrollo.

2.3.1. Symfony

Symfony [22] es un framework de PHP que se utiliza para desarrollar aplicaciones web y que, debido a su arquitectura completamente desacoplada, permite añadir o eliminar Bundles para generar nuevas funcionalidades en la aplicación. Un Bundle es un directorio jerarquizado que contiene todo tipo de archivos. Es como si fuera una librería o un plugin que permite añadir nuevas funcionalidades.

Symfony sigue el patrón de diseño de MVC (Modelo-Vista-Controlador). Este patrón propone la construcción de tres componentes (modelo, vista y controlador) para separar la lógica

de la aplicación de la representación visual. El controlador se encarga de gestionar las peticiones y generar las respuestas pertinentes para cada petición con la información proporcionada por el modelo y la vista. El modelo se encarga de proporcionar los datos necesarios para cada petición que recibe el controlador. Por último, la vista se encarga de proporcionar los elementos visuales para cada petición del usuario. En la figura 2.1 se puede ver el esquema del diseño MVC.

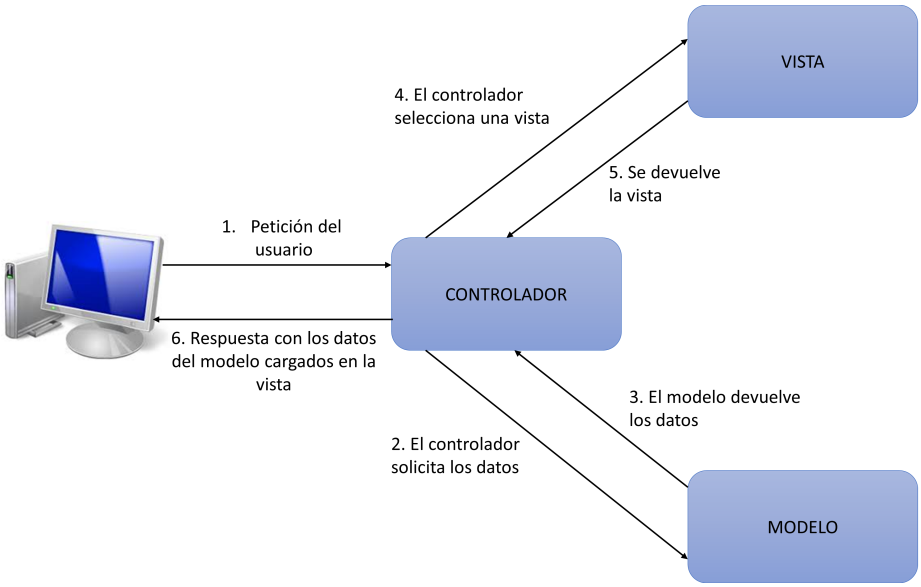

Figura 2.1: Flujo de la información en el patrón MCV
Fuente: elaboración propia

En Symfony, en el modelo se utiliza Doctrine (apartado 2.3.2) para realizar las conexiones con la base de datos. En la vista se usa Twig (apartado 2.3.3) para generar las plantillas. En el Controlador es dónde se ejecuta la lógica de la aplicación y devuelve una repuesta para cada petición.

2.3.2. Doctrine

Doctrine [9] es el ORM (Object Relational Mapper) oficial de Symfony en PHP que nos permite el acceso a la base de datos. Doctrine facilita las conexiones con la base de datos para las consultas a través de objetos, utilizando los mismos conceptos que Hibernate (el ORM de Java) [9].

Se ha seleccionado este ORM ya que es el oficial de Symfony y su uso es sencillo. Además, ofrece algunos comandos para actualizar el esquema de la base de dates sin la necesidad de modificarlo directamente.

2.3.3. Twig

Twig [24] es un motor y lenguaje de plantillas para PHP. Este lenguaje es flexible, rápido y seguro. En Symfony se suele utilizar Twig para crear las plantillas de las vistas.

Se ha utilizado este lenguaje para crear las platillas ya que es muy rápido de codificar y es sencillo de interpretar.

2.3.4. Composer

Composer [5] es un gestor de dependencias en PHP. Permite instalar, actualizar o desinstalar librerías en un proyecto de PHP. Este gestor de dependencias instala las librerías en la carpeta *vendor* del proyecto PHP.

En Symfony, se puede instalar individualmente un bundle, situándose en el directorio raíz, y lanzar el siguiente comando:

```
1 composer install <bundle-name>
```

Otra alternativa es instalar todos los bundles necesarios para el proyecto. Esto se suele hacer la primera vez que se clona el repositorio de un proyecto. Entonces, composer lee del fichero *AppBundle.php* los bundles a instalar. En el repositorio la carpeta vendor, que es donde se instalan los bundles, no se suele subir para ahorrar espacio y aumentar la velocidad de clonación de un proyecto de Symfony. En este caso se utiliza el comando:

```
1 composer install
```

2.3.5. MySQL

MySQL [17] es un sistema de gestión de bases de datos relacional. Es uno de los más populares en el mundo debido a que logra un rápido acceso para una carga veloz de las aplicaciones web que lo utilizan.

El diseño de la base de datos proporcionado utilizaba el lenguaje SQL. Por lo tanto, se ha utilizado en este proyecto.

2.3.6. jQuery

jQuery [13] es una biblioteca de JavaScript rápida, pequeña y con muchas funciones. Permite la manipulación, control de eventos, animación y llamadas Ajax de forma muy sencilla.

Se ha utilizado esta librería para cargar en los formularios algunos datos que, dependiendo del evento que sucede, es necesario mostrar o no cierta información.

2.3.7. MDBootstrap

MDBootstrap (Material Design for Bootstrap) [16] es una framework que incorpora las mejores características de Bootstrap y Material Design.

Bootstrap [4] es el framework de HTML, CSS y Javascript de Twitter. Es el más popular del mercado para desarrollar páginas web de forma responsive, es decir, páginas web que se adaptan a cualquier tamaño de pantalla para poder verse de forma óptima.

Material Design [15] es un lenguaje de diseño desarrollado por Google que trata de seguir los principios de un buen diseño con innovación tecnológica y con el objetivo de hacer diseños más amigables y elegantes.

Se ha utilizado este framework ya que se utiliza en otras aplicaciones de la empresa y para conseguir que la imagen corporativa de las aplicaciones de Cuatroochenta sea la misma.

2.3.8. JSON

JSON (JavaScript Object Notation) [14] es un formato ligero de intercambio de datos. Es un formato de texto que es fácil de leer, escribir y de interpretar.

Este formato se ha utilizado para mostrar en la vista del formulario de oportunidad y en el de añadir un producto a una cuenta, las oportunidades y productos de la base de datos. De esta forma, se puede acceder a ellos utilizando jQuery para cargarlos en los formularios dependiendo el tipo de usuario y de lo que quiere hacer.

2.3.9. PhpStorm

PHPStorm [19] es un IDE (Entorno de Desarrollo Integrado) para PHP desarrollado por JetBrains. Este IDE es uno de los más completos que existen en la actualidad. Además, permite instalar un pluguin ideal para Symfony, que ahorra muchísimo tiempo con las sugerencias mientras se desarrolla el código de cualquier aplicación. Aunque este IDE es de pago, existe una licencia para estudiantes con validez de un año. Como se ha utilizado íntegramente este IDE para el desarrollo de la aplicación, se ha adquirido la licencia de estudiante.

2.3.10. Jira

Jira [12] es una herramienta web que proporciona la gestión y el seguimiento de las tareas de un proyecto. Esta herramienta fue desarrollada por Atlassian. Jira proporciona un entorno para planificar y realizar un seguimiento de los proyectos. En Jira se pueden definir las tareas que se van a llevar a cabo en un proyecto y mientras se están realizando, permite añadir horas de desarrollo en las tareas por parte de los distintos miembros del equipo de desarrollo. Por

lo tanto, se puede obtener un control de horas de los miembros del equipo en el desarrollo del proyecto.

Jira proporciona una interfaz en la que se puede observar a simple vista las tareas que esta llevando a cabo un programador. De esta forma, el programador sabe que tiene que hacer en cada momento. Las tareas suelen ser creadas por el equipo de planificación.

Esta herramienta se ha utilizado para realizar el control de las horas durante la estancia en prácticas y para albergar la pila del producto para planificar en cada sprint cuáles se van a desarrollar.

2.3.11. SourceTree

SourceTree [21] es un cliente GUI (Interfaz Gráfica de Usuario) que permite interactuar con un repositorio de Git o Mercurial para llevar a cabo un control de versiones. Esta herramienta fue desarrollada por Atlassian. SourceTree proporciona una sencilla interfaz con la que se puede llevar a cabo commit, push, pull y merge de los archivos, detectar y resolver conflictos, clonar repositorios, etc. Esta interfaz muestra visualmente las diferentes ramas en las que se está trabajando de una forma muy intuitiva.

Esta herramienta se ha utilizado para controlar el código del CRM juntamente con el repositorio de Bitbucket [3], también desarrollado por Atlassian. Bitbucket es un servicio de alojamiento web para los proyectos que utilizan el sistema de control de revisiones Mercurial y Git.

2.3.12. WampServer

WampServer [25] es una plataforma de desarrollo web para Windows que incorpora Apache, MySQL y PHP. Esta plataforma también incluye phpMyAdmin para gestionar la base de datos. Con esta plataforma, se puede desarrollar una aplicación web de forma local. Se ha utilizado esta plataforma para el desarrollo y la ejecución de la aplicación de CRM que se ha desarrollado.

2.3.13. HeidiSQL

HeidiSQL [11] es una herramienta para gestionar una base de datos en MySQL, Microsoft SQL y PosgreSQL a través de su interfaz. Esta herramienta permite modificar los datos de la base de datos y su estructura de forma remota.

En Cuatroochenta suelen utilizar esta herramienta para gestionar las bases de datos de las aplicaciones que desarrollan. Se utiliza como alternativa a phpMyadmin [18] ya que permite realizar consultas de forma muy rápida.

Capítulo 3

Planificación del proyecto

3.1. Metodología

En Cuatroochenta, la parte previa a la planificación de un proyecto, se lleva a cabo a través del CRM que se ha de desarrollar en este proyecto. El CRM se utiliza para dar de alta las cuentas de los cliente, y se les asigna una oportunidad, que es en sí el proyecto. A continuación, se realiza un seguimiento, para acabar de acordar con el cliente el proyecto, y así poder empezar con el desarrollo.

Una vez han acordado un proyecto, pasa a planificación. En la planificación, se decide cuál va ser el equipo de trabajo y se empieza a hacer los diseños de las interfaces que se usarán para el proyecto.

Por último, se pasa al desarrollo del proyecto. Se utiliza la herramienta de Jira [12] para realizar el seguimiento del proyecto. En esta herramienta se da de alta la oportunidad, que es considerada como el proyecto. Esta oportunidad se compone de incidencias, que son las tareas en las que se desagrega un proyecto. Cada incidencia tiene asignada un responsable de su ejecución. De esta forma, tienen una visión completa de la evolución de los proyectos, y pueden medir la rentabilidad e información del tiempo que les cuesta finalizar las tareas.

Esta metodología que sigue la empresa no es exactamente la metodología de Scrum Manager [23]. No obstante, sí que se asemeja en algunos aspectos y más concretamente en la cultura empresarial subyacente de Cuatroochenta.

Por lo tanto, para este proyecto se ha utilizado la metodología ágil Scrum Manager, apoyándose en Jira para poder realizar la planificación y seguimiento. Al tratarse de un proyecto en el que el participante es el estudiante que ha desarrollado este proyecto, no se tiene en cuenta las reuniones con los otros miembros del equipo. En cambio, sí que se va a detallar la pila del producto, y los sprints seguidos para desarrollar la aplicación web del CRM.

3.2. Planificación inicial

En este apartado se detalla la pila del producto para el desarrollo de la aplicación web de CRM.

La estancia en prácticas tiene una duración total de 300 horas. Estas horas se van a dedicar a las siguientes tareas:

- Formación en Symfony.
- Formación en FOSUserBundle [10] y SgDatatablesBundle [20].
- Formación en el inicio de sesión único (SSO) con cuentas de Google [6] para tratar de desarrollarlo en la aplicación de CRM.
- Análisis del sistema:
  - Elicitación de requisitos.
  - Revisión del sistema en funcionamiento.
- Especificación de la pila del producto.
- Implementación y pruebas de la aplicación de CRM (esta parte se desarrollará mediante la Metodología Scrum Manager).

Por otro parte, en la planificación, no se han incluido las horas dedicadas a las reuniones con el tutor académico para el seguimiento de la estancia en prácticas. Puesto que estas actividades forman parte de las 150 horas dedicadas a la redacción de la memoria del proyecto, consultas con el tutor, etc.

Por último, para el desarrollo del producto, se va a emplear la metodología ágil de Scrum Manager. Para ello, será necesario definir el tamaño de los sprints. Al tratarse de una aplicación que se va a desarrollar por un equipo de trabajo individual, los sprints van a tener una duración de una semana. Además, tampoco se va a seguir las reuniones de Scrum por ser un equipo de desarrollo individual. Por lo tanto, inicialmente se definirá la pila del producto y semanalmente se irán seleccionando las historias de usuario a desarrollar, con la posibilidad de ir modificando conforme a las necesidades que vayan surgiendo durante el desarrollo de la aplicación 480 CRM.

Al tratarse de una metodología ágil, se ha de definir la pila del producto. En lo que resta de sección, se va a detallar la pila del producto de la aplicación de CRM. En Scrum Manager, la pila del producto recoge todas las historias de usuario que se han de implementar en una aplicación. Por lo tanto aquí se pueden observar las tareas que se van a llevar a cabo durante el desarrollo del código de la aplicación. Las historias de usuario están agrupadas en epics definidos inicialmente por temáticas.

En el cuadro 3.1 se puede observar el epic de configuración correspondiente al módulo de configuración de la pila del producto. En este cuadro se utiliza el término de administrador para referirse al administrador global, ya que este tipo de usuario es el que se encarga de modificar

los aspectos de configuración de la aplicación web de CRM. Por otra parte, existe el tipo de usuario gestor de la aplicación que no puede realizar ningún tipo de configuración, pero puede acceder a todas las oportunidades y, en último lugar podemos observar el gestor de oficina que únicamente va a poder interactuar con oportunidades que pertenezcan a su oficina.

Cuadro 3.1: Pila del producto: epic configuración

Historias de usuario del epic configuración			
Id	**Rol**	**Funcionalidad**	**Motivación**
HU01	Como administrador	necesito poder dar de alta, modificar, listar y visualizar las cuentas de los usuarios que utilizarán la aplicación	con la finalidad de poder gestionar por parte de diferentes trabajadores la información relevante para la empresa de los proyectos, clientes y facturación.
HU02	Como administrador	voy a poder modificar, listar y visualizar los roles que van a tener los usuarios de la aplicación	para poder controlar el acceso a los diferentes módulos de la aplicación y permitirles hacer sus labores en función de su rol.
HU03	Como administrador	voy a poder dar de alta, modificar, listar y visualizar los tipos de cuenta de los clientes	con la finalidad de clasificar las cuentas de los clientes.
HU04	Como administrador	necesito poder dar de alta, modificar, listar y visualizar los orígenes de las cuentas de clientes	con la finalidad de poder conocer de dónde vienen mis clientes y tener información para posibles campañas de captación de clientes.
HU05	Como administrador	necesito dar de alta, modificar, listar y visualizar las diferentes áreas de negocio de la empresa	para poder asignar a los trabajadores de un área de negocio, las oportunidades que pertenezcan a esta.
HU06	Como administrador	he de dar de alta, modificar, listar y visualizar las diferentes oficinas de Cuatroochenta	con la finalidad de separar el trabajo de cada oficina.
HU07	Como administrador	voy a poder dar de alta, modificar, listar y visualizar las diferentes monedas con las que opera la empresa	para poder utilizar las monedas pertinentes dentro de las oportunidades de una oficina.
HU08	Como administrador	he de poder dar de alta, modificar, listar y visualizar los diferentes sectores de trabajo	con la finalidad de asignar a las cuentas de los clientes dicho sector y en un futuro poder ofrecerles productos en función de los sectores a los que se dedican.
HU09	Como administrador	voy a poder dar de alta, modificar, listar y visualizar estados de las oportunidades	con la finalidad de tener un control más exhaustivo del estado en el que se encuentra una oportunidad para poder reaccionar a tiempo y ganar una oportunidad de negocio.
HU10	Como administrador	necesito dar de alta, modificar, listar y visualizar los tipos de facturación, tipo de facturas emitidas y tipo de periodicidad de una factura	con el fin de tener más información sobre la facturación de los clientes.
HU11	Como administrador	voy a poder dar de alta, modificar, listar y visualizar los productos que la empresa ofrece	para poder asociar estos productos a las oportunidades y poder detallar todos los productos ofrecidos por la empresa para que los clientes puedan conocerlos.
HU12	Como administrador	voy a poder dar de alta, modificar, listar y visualizar los estados de los productos contratados por los clientes	para tener un control del estado de los servicios prestados a los clientes.

En el Cuadro 3.2 se muestra el epic de oportunidad de la pila del producto, correspondiente al módulo de oportunidades.

Cuadro 3.2: Pila del producto: epic oportunidad

Historias de usuario del epic oportunidades			
HU13	Como administrador global y gestor de la aplicación	necesito poder dar de alta, modificar, listar y visualizar las oportunidades	con la finalidad de poder gestionar su información, para poder realizar un seguimiento de las oportunidades y ver si se puede acordar un proyecto.
HU14	Como gestor de oficina	voy a poder dar de alta, modificar, listar y visualizar una oportunidad que está dentro de la oficina a la que pertenezco	con la finalidad de poder gestionar su información, para poder realizar un seguimiento de las oportunidades de su oficina y ver si se pueden convertir en un proyecto.
HU15	Como administrador global, gestor de la aplicación y gestor de oficina	he de poder asociar productos a una oportunidad	con la finalidad de satisfacer las demandas de los clientes.
HU16	Como administrador global, gestor de la aplicación y gestor de oficina	voy a necesitar proponer presupuestos para las oportunidades juntamente con sus productos	para que el cliente conozca el coste del producto.

En el Cuadro 3.3 se muestra el epic de cuentas de clientes de la pila del producto, correspondiente al módulo de cuentas de clientes.

Cuadro 3.3: Pila del producto: epic de cuentas de clientes

Historias de usuario del epic cuentas de clientes			
Id	**Rol**	**Funcionalidad**	**Motivación**
HU17	Como administrador y gestor de aplicación	he de poder dar de alta, modificar, listar y visualizar las cuentas de los clientes	para gestionar todos los clientes de la empresa y poderles asignar una oportunidad de negocio.
HU18	Como gestor de oficina	necesito dar de alta, modificar, listar y visualizar las cuentas de los clientes que pertenezcan a mi oficina	para tener un control de todos los clientes de mi oficina y poderles asignar oportunidades.
HU19	Como administrador, gestor de aplicación y gestor de oficina	he de poder dar de alta, modificar, listar y visualizar los contactos de una cuenta de un cliente	para saber con quien me puedo poner en contacto con la empresa cliente para solucionar cualquier incidencia.
HU20	Como administrador, gestor de aplicación y gestor de oficina	tengo que poder asignar un contacto como principal	para contactar con él e informarle de cualquier novedad respecto a sus productos o con respecto alguna consulta.

En el cuadro 3.4 se muestra el epic de facturación de la pila del producto, correspondiente al módulo de facturación.

Cuadro 3.4: Pila del producto: epic facturación

Historias de usuario del epic de facturación			
Id	**Rol**	**Funcionalidad**	**Motivación**
HU21	Como administrador y gestor de aplicación	necesito dar de alta, modificar, listar y visualizar las facturas de las oportunidades	para gestionar todos los datos de facturación de una oportunidad.
HU22	Como gestor de oficina	necesito modificar, listar y visualizar las facturas de las oportunidades de mi oficina	para gestionar todos los datos de la facturación de las oportunidades.

En el cuadro 3.4 se muestra el epic de contactos web de la pila del producto, correspondiente al módulo de contactos web.

Cuadro 3.5: Pila del producto: epic contactos web

Historias de usuario del epic de contactos web			
Id	**Rol**	**Funcionalidad**	**Motivación**
HU23	Como administrador y gestor de aplicación	necesito poder dar de alta, modificar, listar y visualizar las cuentas de usuarios procedentes del formulario web de contacto de Cuatroochenta	con la finalidad de poder gestionar la información de los clientes y poder contactar con ellos para tratar de satisfacer sus necesidades.
HU24	Como gestor de oficina	voy a poder modificar, listar y visualizar las cuentas de usuarios procedentes del formulario web de contacto de Cuatroochenta en el que hayan indicado la oficina a la que pertenezco	con la finalidad de poder gestionar la información de los clientes de mi oficina y poder contactar con ellos para tratar de satisfacer sus necesidades.

3.3. Estimación de recursos y costes del proyecto

Los lenguajes que se han utilizado para el desarrollo de la aplicación han sido: PHP 5, HTML 5, Javascript y JQuery. También se ha utilizado MySQL para la base de datos, MDBootstrap para renderizar la vista. Juntamente con el propio framework de Symfony 3.

Para el desarrollo de todo el proyecto se ha utilizado un ordenador portátil del estudiante que ha realizado este proyecto, con el IDE de JetBrains PhpStorm para desarrollar la aplicación. Para poder utilizar este IDE, se ha adquirido una licencia de estudiantes, con validez de un año. También se ha usado la herramienta HeidiSQL para la gestión de la base de datos y Source Tree, un Git GUI para realizar el control de versiones para guardar el repositorio del proyecto en BitBucket.

El control de las horas dedicadas a cada tarea, se ha llevado a cabo con la herramienta de Jira. Esta herramienta es utilizada por todos los trabajadores de Cuatroochenta para la

imputación de las horas de los trabajadores en los diferentes proyectos que está desarrollando la empresa.

Por último, la aplicación se alberga en un servidor web de Amazon Web Services. Más concretamente en Amazon EC2 (Elastic Compute Cloud), con la instancia t2.large [1]. Esta instancia cuenta con dos CPU virtuales y 8 Gb de RAM.

El equipo de trabajo de este proyecto ha estado formado por el autor de esta memoria técnica, que se ha encargado de desarrollar la lógica de negocio del CRM y de un diseñador web que ha llevado a cabo el diseño de las interfaces gráficas.

En el cuadro 3.6 se puede observar el detalle del coste del proyecto en base al equipo de trabajo y a los recursos software y hardware utilizados.

Cuadro 3.6: Costes del proyecto

Coste del proyecto			
Recursos humanos			
Cargo	Precio/hora	Horas	Total
Programador junior	30 €	300	9.000 €
Diseñador web	30 €	50	1.500 €
Software			
Software	Coste Licencia	Duración	Total
JetBrains PhpStorm	0 €	3 meses	0 €
HeidiSQL	0 €	3 meses	0 €
Source Tree	0 €	3 meses	0 €
Jira	7 €	3 meses	21 €
Hardware			
Hardware	Precio/mes	Duración	Total
Servidor Amazon EC2 t2.large	43,75 €	12 meses	525 €
TOTAL			11.046 €

Por lo tanto, el coste del proyecto seria de 11.046 €, pero sin tener en cuenta las horas utilizadas para el diseño de las interfaces es de 9.546 €.

3.4. Seguimiento del proyecto

En esta sección se detalla el seguimiento de la planificación del desarrollo del proyecto.

De las 24 historias de usuarios que se habían de desarrollar, se han descartado dos historias de usuario, HU21 y HU22. Estas pertenecen al epic de facturación que, finalmente se ha prescindido su implementación para la primera versión. Esto se detalla en el apartado 3.4.4.1 Cuarto sprint.

Por otra parte, algunas de las dificultades a la hora de ejecutar las tareas ha sido el diseño de la interfaz, que se modificó durante el desarrollo de la aplicación, y algún error hallado en el esquema de la base de datos, que se tuvo que resolver para seguir con el desarrollo.

3.4.1. Primer sprint

El primer epic que se ha desarrollado ha sido el de oportunidades. Por lo tanto, en este sprint se incluyen algunas historias de usuario relacionadas con la implementación de las tablas de configuración, que son necesarias para poder gestionar una oportunidad. Por ejemplo, es necesario gestionar una oficina, ya que una oportunidad pertenece a una oficina, y se debe indicar la oficina al crear una oportunidad. Por lo tanto, las oficinas deben preexistir.

La motivación para empezar con este sprint ha sido que las oportunidades son el centro de la aplicación de CRM y las que más importancia tendrán.

3.4.1.1. Planificación

En el cuadro 3.7 se pueden ver las historias de usuario que se han planificado para llevarse a cabo durante este sprint.

Cuadro 3.7: Historias de usuario planificadas para el primer sprint

Historias de usuario		
Id	**Rol**	**Funcionalidad**
HU05	Como administrador global	necesito dar de alta, modificar, listar y visualizar las diferentes áreas de negocio de la empresa
HU06	Como administrador global	he de dar de alta, modificar, listar y visualizar las diferentes oficinas de Cuatroochenta
HU07	Como administrador global	voy a poder dar de alta, modificar, listar y visualizar las diferentes monedas con las que opera la empresa
HU08	Como administrador global	he de poder dar de alta, modificar, listar y visualizar los diferentes sectores de trabajo
HU09	Como administrador global	voy a poder dar de alta, modificar, listar y visualizar estados de las oportunidades
HU11	Como administrador global	voy a poder dar de alta, modificar, listar y visualizar los productos que la empresa ofrece
HU12	Como administrador global	voy a poder dar de alta, modificar, listar y visualizar los estados de los productos contratados por los clientes

3.4.1.2. Seguimiento

Durante el primer sprint, se ha llevado a cabo las historias de usuario HU05, HU06, HU07, HU08 y HU09. En las interfaces de estas historias de usuario ha faltado hacer la maquetación, ya que todavía no se habían diseñado las interfaces. Por lo tanto, se ha dejado la lógica implementada a falta de hacer el rediseño.

Durante este sprint, no se ha podido completar la historia de usuario HU10 debido a que el diseño de la base de datos no era el correcto y se ha decidido hacer un rediseño en el esquema de la base de datos, para que tuvieran cabida los requisitos del cliente.

3.4.2. Segundo sprint

En este sprint se ha implementado la historia de usuario que se había planificado en el sprint anterior y, además, se ha empezado a desarrollar el epic de oportunidades, ya que se quería continuar con este epic, debido a su importancia dentro del CRM.

3.4.2.1. Planificación

En el cuadro 3.8 se muestran las historias de usuario que se han planificado para desarrollarse durante el segundo sprint.

Cuadro 3.8: Historias de usuario planificadas para el segundo sprint

Historias de usuario		
Id	**Rol**	**Funcionalidad**
HU11	Como administrador global	voy a poder dar de alta, modificar, listar y visualizar los productos que la empresa ofrece
HU13	Como administrador global y gestor de la aplicación	necesito poder dar de alta, modificar, listar y visualizar las oportunidades
HU14	Como gestor de oficina	voy a poder dar de alta, modificar, listar y visualizar una oportunidad que está dentro de la oficina a la que pertenezco
HU15	Como administrador global, gestor de la aplicación y gestor de oficina	he de poder asociar productos a una oportunidad
HU16	Como administrador global, gestor de la aplicación y gestor de oficina	voy a necesitar proponer presupuestos para las oportunidades juntamente con sus productos

3.4.2.2. Seguimiento

Durante este sprint, se ha podido terminar las HU11, HU13 y HU14. Por lo tanto, han quedado por hacer las HU15 y HU16. Estas son los productos dentro de una oportunidad y la integración con la aplicación de presupuestos.

3.4.3. Tercer sprint

El tercer sprint está centrado en el objetivo de terminar de desarrollar el epic de oportunidades, terminar con las historias de usuario de gestión de tipos de cuents, orígenes de cuentas y tipos de facturación, tipos de facturas emitidas y tipos de periodicidad y aplicar el diseño de las interfaces en el desarrollo hecho hasta ese sprint. De esta forma, el epic de oportunidades que es el principal estaría terminado.

3.4.3.1. Planificación

En el cuadro 3.9 se detallan las historias de usuario que se han planificado en este sprint. La HU25 y HU26 se han añadido a la pila del producto, ya que la lógica de la aplicación estaba implementada, pero faltaba integrar el diseño de las interfaces.

Cuadro 3.9: Historias de usuario planificadas para el tercer sprint

Historias de usuario		
Id	**Rol**	**Funcionalidad**
HU03	Como administrador	voy a poder dar de alta, modificar, listar y visualizar los tipos de cuenta de los clientes
HU04	Como administrador	necesito poder dar de alta, modificar, listar y visualizar los orígenes de las cuentas de clientes
HU10	Como administrador	necesito dar de alta, modificar, listar y visualizar los tipos de facturación, tipo de facturas emitidas y tipo de periodicidad de una factura
HU15	Como administrador global, gestor de la aplicación y gestor de oficina	he de poder asociar productos a una oportunidad
HU25	Como administrador global, gestor de la aplicación y gestor de oficina	he de visualizar el nuevo diseño en el epic de configuración
HU26	Como administrador global, gestor de la aplicación y gestor de oficina	he de visualizar el nuevo diseño en el epic de oportunidad

3.4.3.2. Seguimiento

Las historias de usuario HU03, HU04 y HU15 se han logrado terminar. Respecto al rediseño de las interfaces, se ha podido completar el epic de configuración, HU25. Por lo tanto, faltaba rediseñar el módulo de oportunidades, HU26.

3.4.4. Cuarto sprint

El objetivo del cuarto sprint ha sido terminar el rediseño del epic de oportunidades y el epic de cuentas. Después de las oportunidades, las cuentas es el contenido más importante de 480

CRM.

3.4.4.1. Planificación

En el cuadro 3.10 se detallan las historias de usuario que se han planificado en este cuarto sprint.

Cuadro 3.10: Historias de usuario planificadas para el cuarto sprint

Historias de usuario		
Id	**Rol**	**Funcionalidad**
HU17	Como administrador global	he de poder dar de alta, modificar, listar y visualizar las cuentas de los clientes
HU18	Como gestor de oficina	necesito dar de alta, modificar, listar y visualizar las cuentas de los clientes que pertenezcan a mi oficina
HU19	Como administrador global y gestor de oficina	he de poder dar de alta, modificar, listar y visualizar los contactos de una cuenta de un cliente
HU20	Como administrador global y gestor de	tengo que poder asignar un contacto como principal
HU26	Como administrador global, gestor de la aplicación y gestor de oficina	he de visualizar el nuevo diseño en el epic de oportunidad.

3.4.4.2. Seguimiento

Durante este sprint se ha terminado de integrar el diseño de la interfaz del 480 CRM, HU26. Los nuevos módulos que se iban desarrollando ya integraban este diseño. Además, también se ha acabado de desarrollar el módulo de cuentas, es decir, HU17, HU18, HU19 y HU20. Al finalizar estas historias de usuario, se ha visto que la base de datos no estaba bien mapeada. En las entidades de Symfony, no estaban presentes todas las relaciones de la base de datos, por lo que esta información era inconsistente. Se ha corregido este error.

3.4.5. Quinto sprint

Antes de la planificación de este sprint, ha habido una reunión con el cliente y se ha decidido los módulos que se iban a terminar de desarrollar para una primera versión de 480 CRM. Esta primera versión no iba a contener el módulo de facturación, ni la integración con la aplicación de presupuestos. Tampoco se iba a utilizar la autenticación por SSO de Google, ya que era desconocido por la empresa y sólo se podía hacer con una cuenta de administrador de la G Suite de Google de pago y en este caso, el estudiante no lo era. Por último, se ha acordado que los roles de usuarios no variaran y que será necesaria la configuración de los roles.

Esta primera versión se ha acordado para desarrollarla durante el resto de estancia en prácticas. También se han acordado distintas versiones más avanzadas de la aplicación, para en un futuro desarrollarlas. La especificación de las versiones ha quedado de la siguiente forma:

- Versión 1 de 480 CRM: contiene el módulo de oportunidades, cuentas, contactos web y el módulo de configuración del CRM y de los usuarios. En el módulo de contactos web, se debía de poder editar directamente desde la lista de contactos web, el propietario de este contacto.
- Versión 2 de 480 CRM: contiene los mismos módulos que la versión anterior y se añade la autenticación por SSO de Google.
- Versión 3 de 480 CRM: misma funcionalidad que la versión 2, pero con el módulo de facturación operativo.
- Versión 4 de 480 CRM: igual que la versión 3, pero con el presupuestador integrado.

Por lo tanto, este sprint ha tenido el objetivo de implementar el epic de contactos web y la gestión de usuarios. De esta forma, se ha podido continuar con los propósitos de la primera versión.

3.4.5.1. Planificación

En el cuadro 3.11 se detallan las historias de usuario que se han planificado en este quinto sprint. En HU23 y HU24 se ha añadido la edición del propietario. Además de la pila del producto, se ha eliminado la gestión de roles, HU02.

Cuadro 3.11: Historias de usuario planificadas para el quinto sprint

Historias de usuario		
Id	**Rol**	**Funcionalidad**
HU01	Como administrador global	necesito poder dar de alta, modificar, listar y visualizar las cuentas de los usuarios que utilizarán la aplicación
HU23	Como administrador global y gestor de aplicación	necesito poder dar de alta, modificar, listar (con edición de propietario) y visualizar las cuentas de usuarios procedentes del formulario web de contacto de Cuatroochenta
HU24	Como gestor de oficina	voy a poder modificar, listar (con edición de propietario) y visualizar las cuentas de usuarios procedentes del formulario web de contacto de Cuatroochenta en el que hayan indicado la oficina a la que pertenezco

3.4.5.2. Seguimiento

Durante este sprint, se ha concluido el epic de contactos web, es decir, las HU23 y HU24.

Respecto a la gestión de usuario, no se ha podido terminar la HU01, falta la edición de los datos de un usuario.

3.4.6. Sexto sprint

El objetivo de este sprint ha sido de terminar con la edición de los datos de un usuario, probar el funcionamiento de la aplicación al completo y de publicar la aplicación en un servidor.

3.4.6.1. Planificación

En el cuadro 3.12 se detallan las historias de usuario que se han planificado en este sexto sprint. La HU1 se ha reducido, ya que haba implementado parte de esta historia de usuario en el sprint anterior. Se han añadido dos nuevas historias de usuario para comprobar el funcionamiento completo de la aplicación y para la publicación de la aplicación en un servidor de Amazon.

Cuadro 3.12: Historias de usuario planificadas para el sexto sprint

Historias de usuario		
Id	**Rol**	**Funcionalidad**
HU01	Como administrador global	necesito poder dar de alta, modificar, listar y visualizar las cuentas de los usuarios que utilizarán la aplicación
HU27	Como administrador global, gestor de aplicación y gestor de oficina	voy a usar la aplicación con el funcionamiento esperado
HU28	Como administrador global, gestor de aplicación y gestor de oficina	necesito acceder a la aplicación de forma remota, desde cualquier ordenador con conexión a internet

3.4.6.2. Seguimiento

En este sprint se han podido llevar a cabo las tareas programadas y se ha publicado la aplicación en un servidor de Amazon, tal como se ha comentado.

Capítulo 4

Análisis y diseño del sistema

4.1. Análisis del sistema

En esta sección se va a revisar el análisis del sistema que se ha llevado a cabo durante el proyecto de desarrollo de la aplicación 480 CRM. En primer lugar, se explica la elicitación de requisitos y después se muestra el diagrama de casos de uso de la aplicación. Aunque se haya utilizado Scrum Manager y se haya usado la técnica de las historias de usuario para la especificación de los requisitos funcionales, se ha considerado la realización de este diagrama para ofrecer una visión general de la aplicación y ayuda a comprender las funciones que se pueden llevar a cabo mediante esta aplicación.

4.1.1. Elicitación de requisitos

La elicitación de requisitos es un proceso que se ha llevado a cabo a través de una reunión con el cliente en el cual se ha presentado la aplicación de CRM que utilizaban con anterioridad en la empresa. También se han concretado algunos de estos requisitos por email.

Los principales requisitos que se han acordado tras la observación del sistema en funcionamiento y después de dicha reunión con el cliente han sido los siguientes:

- En el formulario de oportunidad y de producto, se ha de elegir el tipo de producto en función del área de negocio seleccionada.
- En el formulario de oportunidad se ha de mostrar un seleccionable de oportunidad inicial con las oportunidades disponibles de la cuenta seleccionada con anterioridad cuando se selecciona un tipo de producto de mantenimiento.
- En el formulario de oportunidad se ha de preseleccionar el usuario autenticado como propietario y analista de la oportunidad que se está creando a través del formulario.
- En una oportunidad, el campo de fecha se rellenará automáticamente con la fecha actual.

- El código de una oportunidad se asignara un valor autonumérico.
- Se ha acordado un único rol de usuario por usuario (en el CRM anterior había más de uno).
- El las tablas de oportunidades, cuentas y contactos web se debe de poder realizar una búsqueda por filtros.
- En la tabla de contacto web, se podrá modificar el propietario directamente desde la tabla.
- Se ha de utilizar el diseño de las interfaces proporcionado por Cuatroochenta.

A partir de estas necesidades se ha definido el diagrama de casos de uso y a partir de este las historias de usuario que forman la pila del producto.

4.1.2. Diagrama de casos de uso

En la figura 4.1 se observa el diagrama de casos de uso de 480 CRM. En este diagrama no se ha tenido en cuenta la integración de la aplicación de presupuestos y, por lo tanto, no se ha añadido este caso de uso.

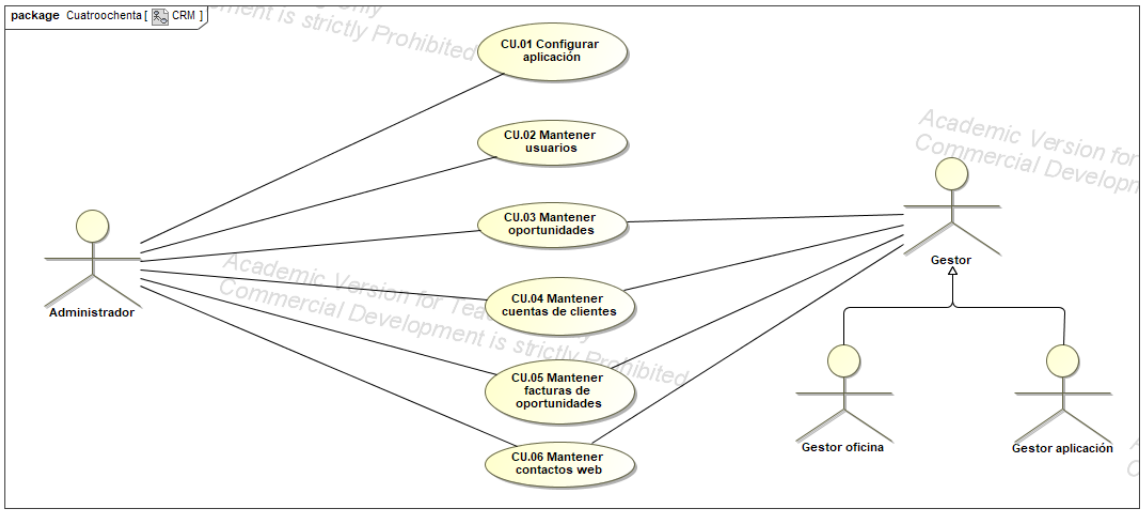

Figura 4.1: Diagrama de casos de uso de la aplicación 480 CRM

Tal como se definió en el alcance organizativo del proyecto en la sección 2.2 de descripción del proyecto que en 480 CRM hay tres tipos de usuarios y se dividen en:

- Usuario administrador global: este tipo de usuarios puede gestionar usuarios y gestionar aspectos de configuración como pueden ser las oficinas que tiene la empresa. Además de estas labores, también puede gestionar oportunidades, cuentas, contactos, contactos web y productos, es decir, va a poder gestionar completamente toda la aplicación de CRM.

- Usuario gestor de aplicación: este tipo de usuario puede gestionar las oportunidades, cuentas, contactos, contactos web y productos ,pero no puede gestionar los usuarios ni los aspectos de configuración.
- Usuario gestor de oficina: este tipo de usuario puede gestionar únicamente oportunidades, cuentas, contactos, contactos web y productos que pertenezcan a la oficina en la que trabaja este tipo de usuario.

Por último, se presenta el cuadro 4.1 que relaciona los casos de uso con las historias de usuario.

Cuadro 4.1: Relación entre casos de uso e historias de usuario

Casos de uso	**Historias de usuario**
CU.01 Configurar aplicación	HU03, HU04, HU05, HU06, HU07, HU08, HU09, HU10, HU11 y HU12
CU.02 Mantener usuarios	HU01 y HU02
CU.03 Mantener oportunidades	HU13, HU14, HU15, y HU16
CU.04 Mantener cuentas de clientes	HU17, HU18, HU19, y HU20
CU.05 Mantener facturas de oportunidades	HU21 y HU22
CU.06 Mantener contactos web	HU23 y HU24

4.2. Diseño de la arquitectura software del sistema

La arquitectura software de la aplicación de 480 CRM es una arquitectura de cliente-servidor. En el servidor se alberga la aplicación de CRM y la base de datos. Los clientes se conectan a la aplicación a través de un navegador web, realizando peticiones al servidor. En la figura 4.2 se muestra un esquema de la arquitectura de cliente-servidor.

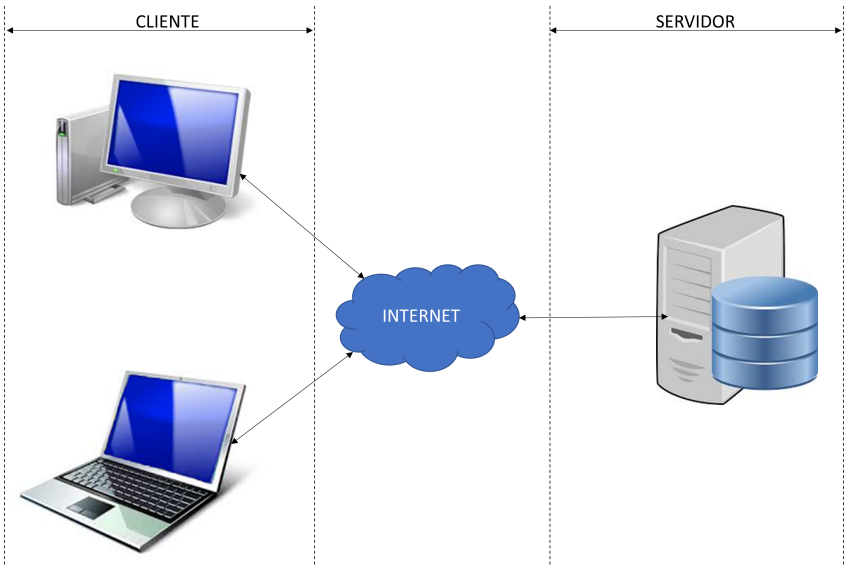

Figura 4.2: Arquitectura cliente-servidor de la aplicación de CRM
Fuente: elaboración propia

Una vez vista la arquitectura de la aplicación, se va a mostrar la estructura que presenta en Symfony esta aplicación.

En la figura 4.3 se puede observar la estructura de los principales directorios de la aplicación 480 CRM. A continuación, se explica brevemente el contenido de cada directorio:

- *config*: contiene los ficheros de configuración de Symfony y de los bundles.
- *views*: contiene las plantillas de Twig que se emplean para el modelado de la vista.
- *translations*: contiene los diccionarios empleados para realizar traducciones.
- *FOSUserBundle*: contiene las plantillas del bundle que han sido rediseñadas para integrar la interfaz de la aplicación. Son las plantillas que están relacionadas con la autenticación, registro y edición de usuarios.
- *AppKernel.php*: fichero que se ejecuta al principio de ejecutar la aplicación y dónde se indican los bundles necesarios, directorio de la cache, de los logs, etc.
- *Controller*: directorio que contiene los controladores de los objetos de la aplicación. El nombre de los controladores suele ser *ObjetoController.php*.
- *Datatables*: directorio que contiene las tablas de los objetos de la aplicación. Son utilizadas por el bundle SgDatatablesBundle para mostrarlas. El nombre de las tablas suele ser *ObjetoDatatable.php*.
- *Entity*: directorio que contiene las entidades de los objetos de la aplicación. El nombre de las entidades suele ser *Objeto.php*.
- *Form*: directorio que contiene los formularios de los objetos de la aplicación. El nombre de estos ficheros suele ser *ObjetType.php*.
- *vendor*: directorio en el que se instalan los bundles de Symfony y los de terceros.
- *css*: directorio que contiene las hojas de estilo propias y las de boostsrap.
- *img*: directorio que contiene las imagenes de la aplicación.
- *js*: directorio que contiene los ficheros de javascript propios y de mdboostsrap.

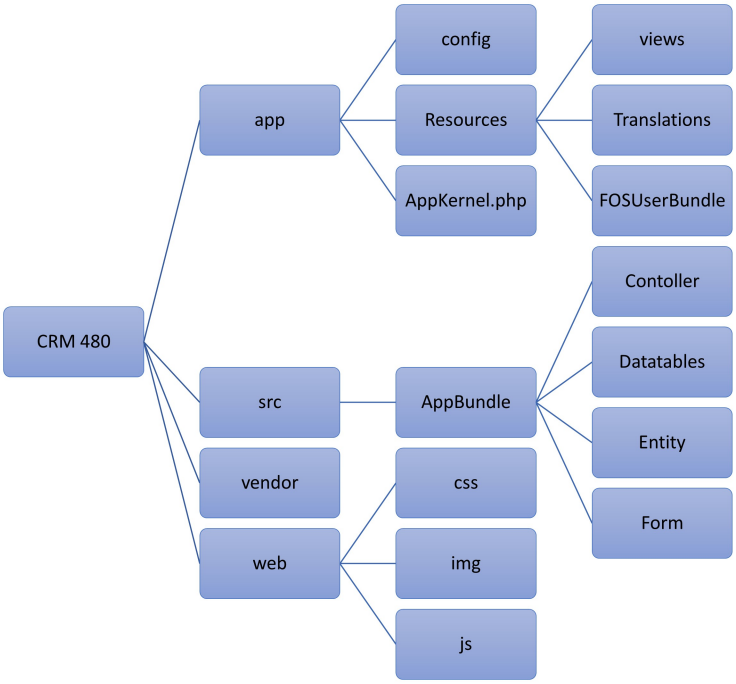

Figura 4.3: Estructura de los directorios principales de 480 CRM

4.3. Diseño de la base de datos

El diseño de la base de datos ha sido proporcionado por Cuatroochenta. Este esquema es el que se utilizaba en la antigua aplicación de CRM. Por lo tanto, ha sido necesario adaptar algunas tablas, ya que algunos requisitos de la aplicación han cambiado.

En este apartado se va a detallar cómo se han llevado a cabo estos cambios en función a los requisitos de la aplicación de 480 CRM. El principal requisito que ha impuesto estos cambios es la clasificación de los tipos de productos en función del área de negocio a la que pertenecen. Esto ocurre cuando se quiere añadir un producto a una cuenta o a una oportunidad, se debe seleccionar en primer momento al área de negocio a la que se va añadir un producto. Entonces, según el área de negocio seleccionada, se puede seleccionar un tipo de producto u otro. Este requisito no estaba presente en la anterior aplicación y se hacía de forma manual.

En la figura 4.4 se puede observar parte del diseño de la base de datos proporcionado por Cuatroochenta del antiguo CRM.

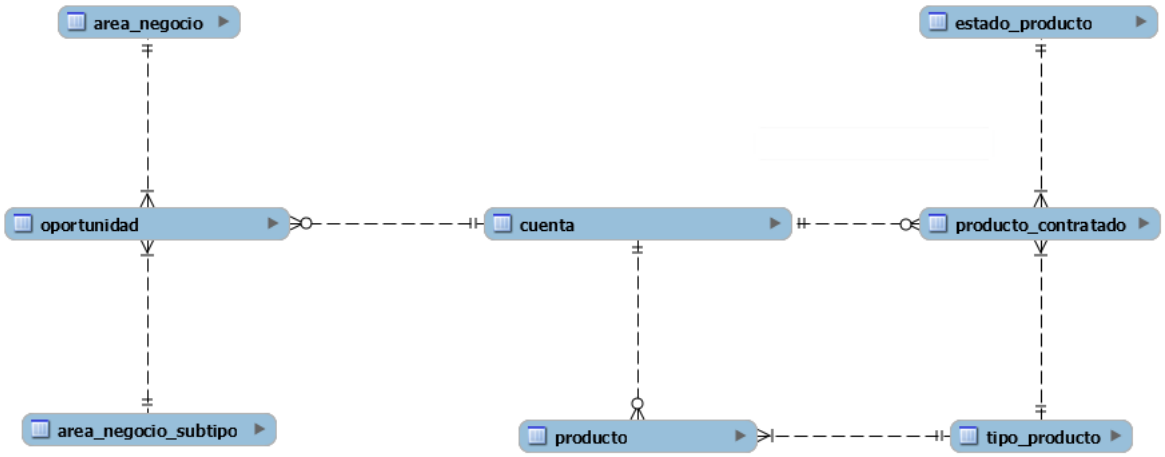

Figura 4.4: Fragmento del esquema de la base de datos de partida

En este diseño, una oportunidad pertenece a una área de negocio. El subtipo de un área de negocio es el producto principal de una oportunidad. Dependiendo el área de negocio a la que pertenece, puede ser uno u otro. Por ejemplo, si el área de negocio es desarrollo app, el subtipo solo puede ser desarrollo app, mantenimiento app y diseño app.

Una cuenta puede puede tener ninguna o varias oportunidades. Cuando tiene una oportunidad, si la oportunidad lo requiere, se pueden añadir más productos y se guardan en la tabla producto, es decir, la tabla producto son los productos secundarios de una oportunidad. La tabla tipo de producto viene a ser lo mismo que el área de negocio subtipo, pero incluyendo el área de negocio. Por lo tanto, la información está replicada.

Por último, una cuenta puede contratar productos independientemente de una oportunidad. Esto suele ocurrir cuando quieren adquirir dominios, licencias de software, etc.

Después de explicar el anterior diseño, se van a enumerar las modificaciones del diseño para dar cabida al requisito de la selección lógica de los productos. En la figura 4.5 se detalla el esquema del nuevo diseño.

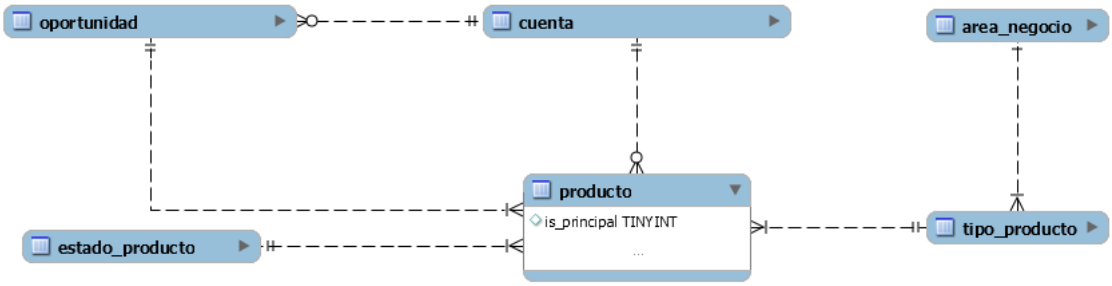

Figura 4.5: Fragmento del esquema de la base de datos mejorado

En el nuevo diseño, una oportunidad está formada por uno o más productos. Va a tener un producto que será el principal y los demás, en caso de haberlos, secundarios. Se ha añadido un

atributo booleano para indicar esta diferencia.

De igual forma que en el diseño anterior, una cuenta puede estar relacionada directamente con productos si los contrata directamente.

La principal diferencia reside en que un producto se clasificará en un tipo de producto y este tipo de producto pertenecerá a una área de negocio en particular.

De esta forma, se cumple el requisito de los tipos de productos y se reduce el número de tablas y la información replicada.

4.4. Diseño de la interfaz

La interfaz de la aplicación ha sido proporcionada por Cuatroochenta. Las plantillas son ficheros en HTML, que contienen las clases de MDBootstrap para saber en la aplicación cuáles había que escoger para seguir con el diseño dado. Cada plantilla se ha tenido que adaptar para poder utilizarla en el CRM con los datos de la base de datos. Se ha tenido que reescribirla en twig.

En el resto de sección, se van a ver las plantillas que se han utilizado para listar las oportunidades, consultar una oportunidad particular y crear una oportunidad nueva. Para el resto de los módulos, el diseño es muy similar.

4.4.1. Listado de oportunidades

En la figura 4.6 se puede observar el diseño de la plantilla para ver todas las oportunidades registradas en el CRM. En todas las plantillas se va a poder ver el menú lateral con los diferentes módulos del sistema, una barra superior en la que se muestra la ruta de en qué lugar del sistema te encuentras y un pequeño menú con los datos del usuario logueado. En el centro de la plantilla es donde va el contenido y es la parte que se va a actualizar. En este caso se puede observar una tabla con todas las oportunidades que hay en el sistema. La cabecera de la tabla permite realizar búsquedas entre los datos de la tabla. En la parte inferior se observa un botón que permite regresar al inicio de la plantilla y, debajo de la tabla, se encuentra la paginación.

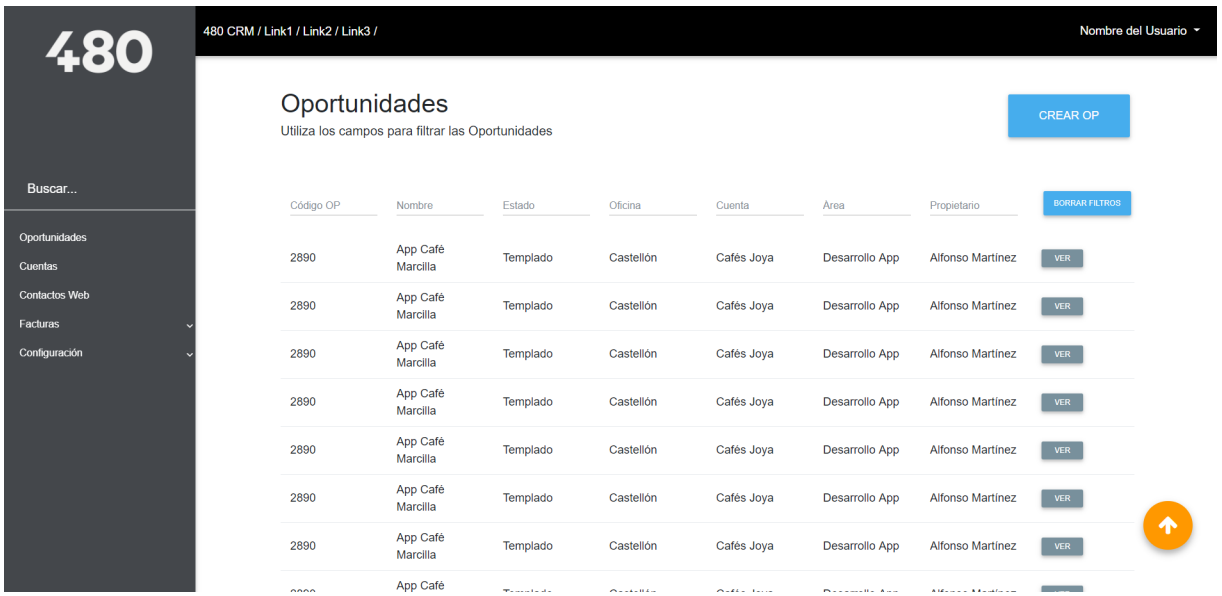

Figura 4.6: Plantilla listado de oportunidades

4.4.2. Mostrar una oportunidad

En la figura 4.7 se puede observar el diseño de la plantilla para acceder a una oportunidad en particular. Este diseño comparte los menús que se han visto en la figura 4.6. En la parte central, se puede contemplar una caja con la información más relevante de una oportunidad, otra caja con los presupuestos de una oportunidad y otra caja con los productos pertenecientes a esa oportunidad.

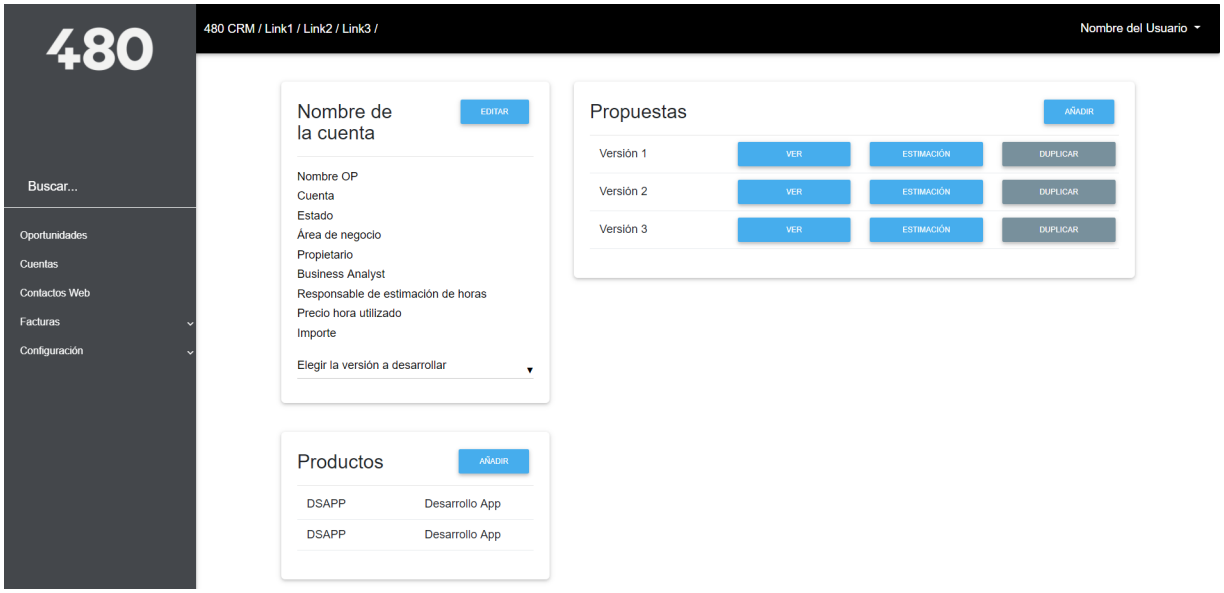

Figura 4.7: Plantilla mostrar una oportunidad

4.4.3. Crear una oportunidad

En la figura 4.8 se puede observar el diseño de la plantilla para crear una oportunidad en particular. Este diseño comparte los menús que se han visto en la figura 4.6. En la parte central se puede ver el formulario de registro para guardar la información de una oportunidad.

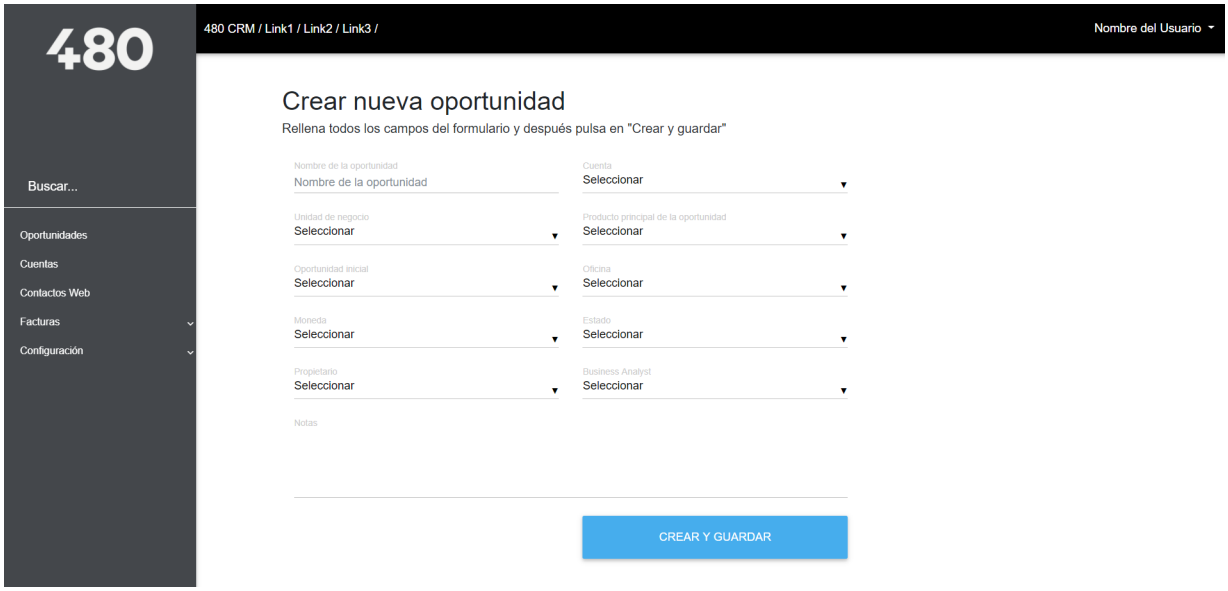

Figura 4.8: Plantilla para crear una oportunidad nueva

Capítulo 5

Implementación y pruebas

En este capítulo se va detallar la implementación de la aplicación de CRM y las pruebas que se han ido haciendo en cada sprint para comprobar que el funcionamiento de la aplicación fuera el deseado. Por lo tanto, este capítulo se divide en seis secciones, correspondientes a cada uno de los sprints realizados.

Antes de empezar con la implementación, se ha preparado el entorno para el desarrollo de la aplicación de CRM.

En primer lugar, se ha importado del dump de la base de datos al servidor local de WampsServer mediante la herramienta HeidiSQL para disponer de la estructura y los datos de la base de datos. Este dump ha sido facilitado por la propia empresa. En un principio compartía la misma estructura que el CRM que utilizaban con anterioridad. Sin embargo, durante la implementación, se ha ido modificando para adaptarlo a algunos requisitos.

Después, se ha utilizado la herramienta de SourceTree para clonar el repositorio donde estaba el código con las entidades de la base de datos. Una entidad es un objeto PHP que representa una tabla de la base de datos. A través de anotaciones, se puede definir las relaciones que presenta con otras tablas y el tipo de datos de los atributos. En el código 5.1 se puede observar el código de la entidad sector de trabajo de una cuenta con ejemplos de anotaciones. Por ejemplo, se utiliza la anotación *@ORM\Id* para indicar que es la clave primaria.

Código 5.1: Código de la entidad de sector de trabajo de una cuenta

```
1  class Sector
2  {
3      /**
4       * @ORM\Id
5       * @ORM\Column(type="bigint", length=20)
6       * @ORM\GeneratedValue(strategy="AUTO")
7       */
8      protected $id;
9
10     /**
11      * @ORM\Column(type="string", length=255, nullable=true)
12      */
```

```
13      protected $nombre;
14
15      /**
16       * @return mixed
17       */
18      public function getId()
19      {
20          return $this->id;
21      }
22
23      /**
24       * @param mixed $id
25       */
26      public function setId($id)
27      {
28          $this->id = $id;
29      }
30
31      /**
32       * @return mixed
33       */
34      public function getNombre()
35      {
36          return $this->nombre;
37      }
38
39      /**
40       * @param mixed $nombre
41       */
42      public function setNombre($nombre)
43      {
44          $this->nombre = $nombre;
45      }
46  }
```

Por último, se ha utilizado Composer para instalar todos los bundles necesarios para utilizar la aplicación. Los bundles se instalan en la carpeta *vendor*. Es una buena práctica no subir esta carpeta en el repositorio para que no ocupe tanto espacio. Por lo tanto, solo es necesario instalar los bundles la primera vez que se clona un repositorio.

Una vez se ha finalizado la preparación del entrono de desarrollo, se ha pasado a la implementación de las distintas historia siguiendo la metodología Scrum y por lo tanto ejecutando sucesivos sprints.

Antes de empezar, cabe mencionar que la aplicación se ha desarrollado en español e inglés con el bundle BazingJsTranslationBundle [2]. Para ello, se ha ido construyendo un diccionario con las palabras que se iban a traducir. Cada vez que se ha querido visualizar algún tipo de texto, se ha hecho a través de este traductor. Únicamente se ha traducido el esqueleto de la aplicación, pero no el contenido de la base de datos.

5.1. Primer sprint

En el seguimiento del primer sprint (apartado 3.4.1.1) se ha expuesto que las historias de usuario que se han llevado a cabo han sido las relacionadas con la implementación de las tablas de configuración que han sido necesarias para poder gestionar una oportunidad. Estas se pueden resumir en la gestión del área de negocio, oficina, moneda, sector, estado de la oportunidad y estado del producto.

5.1.1. Detalles de implementación

Antes de empezar con la implementación, se ha instalado y configurado el bundle SgDatatablesBundle en base a la documentación de este bundle.

Para cada objeto que se ha desarrollado durante este sprint, se ha implementado el controlador, un formulario, la tabla para SgDatatablesBundle y las plantillas.

En el controlador se han creado los métodos de listar, añadir, borrar y editar para poder gestionar los objetos. Para el listado se ha utilizado las tablas del bundle. Para añadir y editar se ha usado el formulario. Para todos los métodos se han empleado las plantillas de twig para mostrar la interfaz de la aplicación.

Todos los controladores se han implementado con estos métodos base. Según las necesidades se han añadido nuevos, en este caso con estos basta. La edición y visionado de objetos se ha considerado como edición. Para la gestión de los objetos, en el controlador se ha utilizado el *EntityManager* de Doctrine. Por defecto, en Symfony, los controladores contienen este objeto al extender la clase Controller. Por lo tanto, a través del *EntityManager* se ha podido modificar los registros de la base de datos u obtener todos los registros.

Para la implementación de los formularios se han añadido todos los atributos que conforman la entidad que se ha estado desarrollando. Por ejemplo, para la entidad Sector, solo va a contener el nombre, ya que está formado además por el identificador, pero que al ser auto numérico no se debe de modificar y por eso no se añade.

Para las tablas del bundle, se han añadido los campos que se deseaban mostrar. Además, se ha añadido los botones de edición y borrado. En el de edición, el controlador muestra la vista del formulario, mientras que en el de borrado espera confirmación para realizar la acción. Cuando se ha borrado un objeto, en la base de datos se marca como que está borrado, es decir, no se elimina de la base de datos.

El código de este bundle se puede ver con más detalle en código 5.2. En las líneas 13-25 se configura la tabla. Las tablas que se han implementado con este bundle permiten hacer un filtrado de sus datos a través de las cabeceras de la tabla. En las líneas 28-37 se añaden los campos que se van a mostrar y en las líneas 38-69 se añaden los botones de editar y eliminar para cada registro de la tabla.

Código 5.2: Código de SectorDatatble.php

```php
/**
 * Class SectorDatatable
 * @package AppBundle\Datatables
 */
class SectorDatatable extends AbstractDatatable
{
    /**
     * {@inheritdoc}
     */
    public function buildDatatable(array $options = array())
    {
        $this->language->set(array(
            'cdn_language_by_locale' => true
        ));

        $this->options->set(array(
            'individual_filtering' => true,
            'individual_filtering_position' => 'head',
            'order_cells_top' => true,
            'order_multi' => true,
            'classes' => 'table table-hover',
            'page_length' => 20,
            'paging_type' => Style::FULL_NUMBERS_PAGINATION,
        ));

        $this->columnBuilder
            // Add new columns headers
            ->add('id', Column::class, array(
                'title' => '#',
                'searchable' => false,
                'visible' => false,
                'orderable' => true,
                ))
            ->add('nombre', Column::class, array(
                'title' =>  $this->translator->trans('TR_NAME'),
                ))
            ->add(null, ActionColumn::class, array(
                'title' => $this->translator->trans('TR_ACTIONS'),
                'actions' => array(
                    // edit button
                    array(
                        'route' => 'sector_edit',
                        'route_parameters' => array(
                            'sector' => 'id'
                        ),
                        'label' => $this->translator->trans('TR_EDIT'),
                        'attributes' => array(
                            'rel' => 'tooltip',
                            'title' => $this->translator->trans('sg.
                                datatables.actions.edit'),
                            'class' => 'btn btn-sm btn-blue-grey py-1
                                px-2 waves-effect waves-light',
                            'role' => 'button'
                        ),
                    ),
```

```
54                      // delete button
55                      array(
56                          'route' => 'sector_delete',
57                          'route_parameters' => array(
58                              'sector' => 'id'
59                          ),
60                          'label' => $this->translator->trans('TR_DELETE'
                                ),
61                          'confirm' => true,
62                          'confirm_message' => $this->translator->trans('
                                TR_CONFIRMING'),
63                          'attributes' => array(
64                              'rel' => 'tooltip',
65                              'class' => 'btn btn-sm btn-blue-grey py-1
                                    px-2 waves-effect waves-light',
66                              'role' => 'button',
67                          ),
68                      )
69                  )
70              ))
71          ;
72      }
73
74      /**
75       * {@inheritdoc}
76       */
77      public function getEntity()
78      {
79          return 'AppBundle\Entity\Sector';
80      }
81
82      /**
83       * {@inheritdoc}
84       */
85      public function getName()
86      {
87          return 'sector_datatable';
88      }
89  }
```

Por último, se han implementado las plantillas en twig para las vistas. Se ha creado un fichero base con el menú lateral y superior. Cada plantilla comparte esta base y se modificará el contenido. En el 5.3 se puede observar el código de la plantilla para crear un nuevo sector. En la primera línea se extiende la base. El bloque de *breadcumb* y *body* se sobrescriben en el fichero base. En la línea 20, se añade el formulario de un sector.

Código 5.3: Código de la plantilla para añadir un nuevo sector

```
1  {% extends 'base.html.twig' %}
2  {% block breadcrumb %}
3      <ol class="breadcrumb">
4          <li class="breadcrumb-item">{{ "TR_CONFIGURATION" | trans }}</a
               ></li>
5          <li class="breadcrumb-item"><a href="{{ path('sector_index') }}
```

```
            ">{{ "TR_SECTORS" | trans }}</a></li>
6           <li class="breadcrumb-item active" aria-current="page">{{ "
                TR_NEW_M" | trans }}</li>
7       </ol>
8   {% endblock %}
9
10  {% block body %}
11      <div class="row mb-3">
12          <div class="col-sm-12">
13              <h2>{{ "TR_SECTOR_NEW" | trans }}</h2>
14              <h6>{{ "TR_SAVE_DESCRIPTION" | trans }}</>
15          </div>
16      </div>
17      <div class="container-fluid mt-3">
18          <div class="row">
19              <div class="col-sm-12 col-lg-8">
20                  {% include ':sector:form.html.twig' with {'form': form}
                        %}
21              </div>
22          </div>
23      </div>
24  {% endblock %}
```

5.1.2. Pruebas

A medida que se ha ido creando el controlador, el formulario, la tabla para SgDatatablesBundle y las plantillas para un objeto, se han ido haciendo algunas pruebas.

Al tratarse de objetos que se componen por un auto numérico y un nombre, se ha comprobado que la inserción, la edición y el borrado de un objeto fuera el esperado. Además, se ha comprobado que al borrar un objeto pidiese confirmación y que, posteriormente, no se mostrara en la tabla de los objetos.

5.2. Segundo sprint

En el seguimiento del segundo sprint (apartado 3.4.2.1) se ha explicado qué historias de usuario se han llevado a cabo. En este caso ha sido la de gestionar productos y las del epic de oportunidades.

5.2.1. Detalles de implementación

En primer lugar, se ha terminado de implementar la gestión de un producto. Se ha creado el controlador, el formulario, la tabla y las plantillas de la vista.

En segundo lugar, se ha configurado e instalado el bundle de FOSUserBundle. En la confi-

guración del bundle, se han modificado las plantillas del bundle para que se adaptaran al diseño de las interfaces. Esto se ha hecho extendiendo la plantilla base en las plantillas del bundle. En en directorio *app\Resources* se han creado las plantillas que va a sobrescribir Symfony. Por defecto el bundle contiene sus plantillas, pero si en esta ubicación encuentra alguna plantilla que coincide con el nombre, la carga y la muestra.

Después, se ha empezado con el desarrollo de la gestión de oportunidades. En primer lugar, se ha implementado el controlador con las acciones básicas de listar, crear, editar, borrar y mostrar. En este caso sí que se va a mostrar una oportunidad, ya que es se muestra información de sus productos. Además, cuando se añade una oportunidad, se ha añadido en el controlador el código para crear el producto en función de los datos del formulario, ya que en el diseño de la base de datos se ha visto que la oportunidad y el producto son tablas diferentes.

En el código 5.4 se puede observar la implementación del método del controlador que permite mostrar la tabla con las oportunidades. En primer lugar, se comprueba el rol del usuario. Si es un administrador global o gestor de aplicación se cargan todas las oportunidades, en cambio si es un gestor de oficina, se cargan las oportunidades de su oficina. Se le pasa esta consulta al bundle y se encarga de gestionarla. Una vez cargados los datos en la tabla, el controlador devuelve la vista de la tabla con las oportunidades.

Código 5.4: Acción del controlador de una oportunidad que muestra la tabla con todas las oportunidades dependiendo del rol del usuario

```
1
2  public function indexAction(Request $request)
3  {
4      $user = $this->getUser();
5      $role = $user->getRole()->getId();
6      $datatable = $this->get('sg_datatables.factory')->create(
           OportunidadDatatable::class);
7
8      $options['container'] = $this->container;
9      $options['user'] = $user;
10
11     $datatable->buildDatatable($options);
12     $isAjax = $request->isXmlHttpRequest();
13
14     if($role == Role::GESTOR_OFICINA){
15         if ($isAjax) {
16             $oficina = $user->getOficina();
17             $responseService = $this->get('sg_datatables.response');
18             $responseService->setDatatable($datatable);
19             $responseService->getDatatableQueryBuilder();
20
21             $datatableQueryBuilder = $responseService->
                   getDatatableQueryBuilder();
22
23             $qb = $datatableQueryBuilder->getQb();
24             $qb->andWhere('oficina = (:oficina)');
25             $qb->setParameter('oficina', $oficina);
26
27             return $responseService->getResponse();
28         }
```

```
29      }
30      else{
31          if ($isAjax) {
32              $responseService = $this->get('sg_datatables.response');
33              $responseService->setDatatable($datatable);
34              $responseService->getDatatableQueryBuilder();
35
36              return $responseService->getResponse();
37          }
38      }
39      return $this->render('oportunidad/index.html.twig', array(
40          'datatable' => $datatable,
41      ));
42  }
```

A continuación, se ha creado la tabla de las oportunidades. En este caso, se han mostrado los atributos principales de las oportunidades. En esta tabla, se ha añadido un botón que vacía todos los filtrados y actualiza los valores de la tabla a través de jQuery.

Después, se ha creado el formulario para insertar y modificar una oportunidad. Al crear una oportunidad, se debe satisfacer el requisito de que el tipo de producto principal de una oportunidad se debe de seleccionar en función del área de negocio. Además, existe otro requisito y es que si el tipo de producto es de mantenimiento, quiere decir que la oportunidad tiene una oportunidad predecesora. Por lo tanto, se debe permitir seleccionar una oportunidad de la misma cuenta a la que se marca como oportunidad inicial. Por último, se ha implementado el requisito de preseleccionar el propietario y analista de una oportunidad con el usuario autenticado en el sistema.

Para satisfacer el último requisito, el controlador selecciona el propietario y el analista como el usuario que está autenticado en la aplicación y, después, muestra la vista del formulario.

Para poder satisfacer los otros requisitos, se ha seguido una estrategia similar. El controlador envía todos los tipos de productos (los tipos de productos tienen un campo que indica el área de negocio a la que pertenecen) y las oportunidades a la vista. Entonces, la vista codifica los datos en formato JSON y jQuery se encarga de alterar los seleccionables de los productos y mostrar un seleccionable de oportunidad inicial en función del área de negocio y del tipo de producto que el usuario selecciona. El código en jQuery se detalla en el código 5.5.

La función *refreshProducto()* actualiza los tipos de producto que se pueden elegir. Esta función se ejecuta cuando el usuario modifica la unidad de negocio. Esta función lee el área de negocio que ha elegido el usuario en el formulario y recarga el listado del tipo de producto con los valores del JSON, *myData*, que comparten el área de negocio seleccionada por el usuario. Como se ha utilizado mdboostrap, es necesario reiniciar el material select para actualizar los seleccionables. El material select permite mostrar el desplegable del seleccionable de una forma agradable para el usuario. Además de esta función, cuando se modifica el área de negocio, se oculta, si esta visible, el seleccionable de oportunidad inicial.

La función *refreshOportunidadInicial()* actualiza el seleccionable de oportunidades inicial si el tipo de producto es de mantenimiento. Si no lo es, oculta este seleccionable.

La función *tryShowOportunidadInicial()* muestra el seleccionable de oportunidad. Esta función se lleva a cabo cuando el usuario elige la cuenta de un cliente y actualiza las oportunidades iniciales a mostrar de la cuenta seleccionada por el usuario. El proceso es muy similar al de la función *refreshProducto()*.

Código 5.5: Código jQuery que recarga los seleccionables del formulario de una oportunidad en función de los valores que toman el área de negocio y el tipo de producto

```
1  function refreshProducto(){
2      var area_negocio_id = $(".unidad-negocio-select:not(.select-wrapper
           )").val();
3      var tipos_producto = $(".tipo-producto-select:not(.select-wrapper)"
           );
4      $(".tipo-producto-select:not(.select-wrapper)").empty();
5      tipos_producto.append('<option value="" disabled selected>-</option
           >');
6
7      var myData = $('#tipos-productos').data('tipos_productos');
8      var myOptions = myData.map(function(element){
9          if(element.area_negocio.id == area_negocio_id){
10             return '<option value="' + element.id + '">' + element.
                   nombre + '</option>';
11         }
12         return ''
13     }).join('');
14
15     tipos_producto.append(myOptions);
16     tipos_producto.material_select('destroy');
17     tipos_producto.material_select();
18 }
19
20 function refreshOportunidadInicial(){
21     var cuenta_id = $("#appbundle_oportunidad_cuenta").val();
22     var oportunidad_inicial = $(".oportunidad-inicial-select:not(.
           select-wrapper)");
23
24     oportunidad_inicial.empty();
25     oportunidad_inicial.append('<option value="" disabled selected>-</
           option>');
26
27     var myData = $('#oportunidades').data('oportunidades');
28     var myOptions = myData.map(function(element){
29         if(element.cuenta.id == cuenta_id){
30             return '<option value="' + element.id + '">' + 'C.OP ' +
                   element.codigo_op + ': ' + element.nombre + '</option>'
                   ;
31         }
32         return ''
33     }).join('');
34
35     oportunidad_inicial.append(myOptions);
36     oportunidad_inicial.material_select('destroy');
37     oportunidad_inicial.material_select();
38 }
39
```

```
40 function tryShowOportunidadInicial() {
41     var tipo_producto = $(".tipo-producto-select.select-wrapper").find(
           '.active.selected').text();
42     if (tipo_producto.toUpperCase().startsWith('MANTENIMIENTO') ||
           tipo_producto.toUpperCase().startsWith('MAINTENANCE')) {
43         $(".hide").removeClass("hide");
44     }
45     else {
46         hideOportunidadInicial();
47     }
48 }
49
50 function hideOportunidadInicial(){
51     $(".oportunidad-inicial").removeClass("hide");
52     $(".oportunidad-inicial").addClass("hide");
53 }
54
55 jQuery(document).ready(function() {
56     refreshProducto();
57     refreshOportunidadInicial();
58
59     //Cambio unidad de negocio-> cambio los productos
60     $(".unidad-negocio-select.select-wrapper").on('change', function(){
61         refreshProducto();
62         hideOportunidadInicial();
63     });
64     //Cambio el tipo de producto->muestro op.inicial si es de
           mantenimiento
65     $(".tipo-producto-select").on('change', function(){
66         tryShowOportunidadInicial();
67     });
68     //Cambio la cuenta-> recargo oportunidades iniciales
69     $("#appbundle_oportunidad_cuenta").on('change', function(){
70         refreshOportunidadInicial();
71     });
72  });
```

5.2.2. Pruebas

En la gestión de los productos, se ha comprobado que la inserción, la edición y el borrado de un objeto fuera el esperado. Además, se ha comprobado que al borrar un objeto pidiese confirmación y que, posteriormente, no se muestra en la tabla de los objetos.

En las oportunidades, se han llevado a cabo más comprobaciones. En el formulario para añadir una oportunidad, se han hecho las siguientes pruebas y se han superado satisfactoriamente:

- Se ha probado a dejarse campos vacíos del formulario. El resultado ha sido que únicamente permite enviar el formulario si se deja el campo de notas vacío.
- El formulario no deja introducir texto en atributos numéricos.

- Al crear el formulario, está seleccionado el usuario autenticado en la aplicación como analista y propietario.
- El formulario se actualiza correctamente al modificar el área de negocio.
- El formulario se actualiza correctamente al modificar el tipo de producto. Cuando el tipo producto es de mantenimiento, se muestra el seleccionable de oportunidad inicial con los posibles valores correctos.
- El formulario se actualiza correctamente al modificar la cuenta del cliente. Además, el seleccionable de oportunidad inicial tiene oportunidades de la cuenta seleccionada.

En la tabla de oportunidades, se han llevado a cabo las siguientes pruebas y se han superado correctamente:

- Al acceder como administrador global en la aplicación se muestran todas las oportunidades de la aplicación.
- Al acceder como gestor de aplicación en la aplicación se muestran todas las oportunidades de la aplicación.
- Al acceder como gestor de oficina en la aplicación se muestran todas las oportunidades de su oficina. Esto se ha probado con diferentes usuarios de diferentes oficinas.
- El filtrado individual de cada columna funciona correctamente.
- El filtrado múltiple filtra correctamente las oportunidades.
- El botón de borrar filtros elimina los filtros y actualiza los datos de la tabla.
- La paginación de la tabla permite desplazarse por todas las oportunidades de la aplicación.
- El botón para ver muestra correctamente los datos de la oportunidad seleccionada.

5.3. Tercer sprint

En el seguimiento del tercer sprint (apartado 3.4.3.1) se ha detallado las historias de usuarios que se han implementado durante este sprint. En concreto, han sido para terminar con el epic de oportunidades y de configuración, a excepción de la gestión de usuarios y roles, y aplicar el diseño en las interfaces de estos módulos.

5.3.1. Detalles de implementación

En primer lugar, se ha terminado de desarrollar la funcionalidad que no se había podido completar durante el sprint anterior, es decir, poder añadir nuevos productos dentro de una oportunidad. Un ejemplo podría ser que para una oportunidad de un desarrollo app, el cliente quiera añadir el producto de marketing digital que ofrece Cuatroochenta.

Para desarrollar esta funcionalidad, se ha creado un método en el controlador de que mostrara el formulario con la información de un producto que debe rellenar el usuario. En este formulario, también se debía mostrar un tipo de producto en función al área de negocio seleccionada. Por lo tanto, se ha utilizado parte del código 5.5 para conseguir este resultado.

En el código 5.6 se muestra cómo se puede añadir un atributo seleccionable al formulario, concretamente, para seleccionar a que área de negocio pertenece el producto. Esto se hace de forma muy sencilla diciendo que es del tipo *EntityType* y la clase a la que hace referencia, *AppBundle:AreaNegocio*.

Código 5.6: Parte del código del formulario del producto, en el que se detalla como añadir en el formulario la relación con otra tabla. Concretemente con el área de negocio.

```
1  class ProductoType extends AbstractType
2  {
3      public function buildForm(FormBuilderInterface $builder, array
           $options)
4      {
5
6      ...
7
8       $builder
9
10     ->add('area_negocio', EntityType::class,
11                  array(
12                      'class' => 'AppBundle:AreadNegocio',
13                      'choice_label' => 'nombre',
14                      'label'  => 'TR_BUSINESS_UNIT',
15                      'attr'=> array('class'=>'mdb-select unidad-negocio-
                            select card-text form-select'),
16                  )
17              )
18       }
19       ...
20 }
```

Una vez terminado este módulo, se ha pasado a desarrollar las historias de usuario que faltaban del epic de configuración, tipos de cliente, origen de cuenta y facturación facturación, tipo de facturas emitidas y tipo de periodicidad de una factura.

Para cada uno de estos objetos, se ha desarrollado el controlador base con las acciones de listar, añadir, editar y borrar. También se ha creado las tablas y los formularios. Las plantillas para cada vista se han creado con el diseño. Esto se ha logrado añadiendo los elementos html con las clases necesarias e imitando la estructura que presentan los ficheros del diseño. De esta forma la aplicación se visualiza igual que en los diseños.

Después, se ha integrado el diseño en las historias de usuario que se habían desarrollado en el primer sprint y en la gestión de productos.

5.3.2. Pruebas

En la gestión de los tipos de cuenta, orígenes de cuentas y tipos de facturación, tipos de facturas emitidas y tipo de periodicidad, se ha comprobado que la inserción, la edición y el borrado de un objeto fuera el esperado. Además, se ha comprobado que al borrar un objeto pidiese confirmación y que, posteriormente, no se mostrara en la tabla de los objetos.

También se ha comprobado de forma visual, que el diseño en el módulo de oportunidades fuera el mismo que el proporcionado por la empresa.

En el formulario de añadir un producto a una oportunidad se han realizado las siguientes pruebas:

- Se ha probado dejar campos vacíos del formulario. El resultado ha sido que únicamente permite enviar el formulario si se deja el campo de notas vacío.
- El formulario no deja introducir texto en atributos numéricos.
- El formulario se actualiza correctamente al modificar el área de negocio, modificando los tipos de productos disponibles del seleccionable de tipo de producto.

5.4. Cuarto sprint

En el seguimiento del cuarto sprint (apartado 3.4.4.1) se ha mostrado las historias de usuarios que se han implementado durante este sprint. Se pueden resumir en la maquetación del módulo de oportunidades con el diseño proporcionado y la implementación de módulo de cuentas. En concreto, han sido para terminar con el epic de oportunidades y de configuración, a excepción de la gestión de usuarios y roles, y aplicar el diseño en las interfaces de estos módulos.

5.4.1. Detalles de implementación

En primer lugar, se ha terminado de integrar el diseño en el módulo de oportunidades. Se ha conseguido añadiendo los elementos html con las clases necesarias e imitando la estructura que presentan los ficheros del diseño. Además, se ha modificado el método del controlador para que enviara a la plantilla de la vista todos los productos de una oportunidad. En el código 5.7 se detalla cómo se ha creado la tabla que muestra todos los productos de una oportunidad. Se recorren todos los productos y se van añadiendo en la tabla. El producto principal no se expone ya que se detalla con el resumen de los datos de la oportunidad. En esta tabla se ha añadido un botón para ver los detalles del producto.

Código 5.7: Parte del código de la plantilla de twig que se utiliza para mostrar una oportunidad

```
1 <table class="table table-hover">
2  <tbody>
3     {% for producto in productos %}
4      <tr>
```

```
5         <th scope="row">
6             {{ producto.tipoProducto.codigo }}
7           </th>
8           <td>
9               {{ producto.tipoProducto.nombre }}
10          </td>
11          <td>
12               <a href="{{ path('oportunidad_producto_show', {'
                     oportunidad': oportunidad.id, 'producto': producto.id
                     }) }}" class="btn btn-sm btn-blue-grey btn-block px-3
                      py-1">
13                {{ "TR_SHOW" | trans }}
14               </a>
15          </td>
16        </tr>
17       {% endfor %}
18       </tbody>
19  </table>
```

En segundo lugar, se ha desarrollado el epic de cuentas de clientes. Se ha implementado el controlador base, la tabla del bundle con los atributos a mostrar deseados (estos se han sacado del diseño de la interfaz de esta tabla), el formulario y las plantillas de la vista.

En el controlador de productos se ha añadido un método para mostrar el formulario de producto, cuando se desee añadir un producto directamente a una cuenta. En la vista de una cuenta se muestra una tabla con los productos asociados directamente con una cuenta. En esta tabla se muestra el botón para añadir los productos. Esta tabla es muy parecida a la tabla de los productos de una oportunidad. El formulario que se utiliza para añadir un producto a una cuenta, también utiliza parte del código jQuery para actualizar los tipos de producto en función del área de negocio seleccionada.

En el controlador de contactos se ha añadido un método para mostrar el formulario de un contacto, cuando se desee añadir un contacto en una cuenta. En la vista de una cuenta, se muestra la tabla que contiene todos los contactos de una cuenta. Además, muestra cuál es el principal. En esta tabla se muestra el botón para añadir los contactos a la cuenta, siendo esta muy parecida a la tabla anterior.

En el código 5.8 se detalla parte del código para añadir un contacto. Al añadir un contacto y lo marco como principal, el controlador guarda este contacto como principal y borrará el contacto principal que previamente tuviera almacenado en esta cuenta. En cambio, si no marco el contacto como principal, el controlador comprueba que haya algún contacto principal, en caso afirmativo, inserta este nuevo contacto. En caso negativo, lo guarda como principal. ya que en una cuenta debe de existir un contacto principal. En las líneas 6-10 se puede observar lo fácil que es con Doctrine obtener los objetos de la base de datos.

Código 5.8: Parte del código del controlador de contactos que modifica el contacto principal al añadir un nuevo contacto

```
1  ...
2  if ($form->isSubmitted() && $form->isValid()) {
3   $em = $this->getDoctrine()->getManager();
4      //Si anyado un contacto principal borro el otro contacto principal
```

```
5       if($contacto_cuenta->getPrincipal()){
6        $contacto_cuenta_repo = $em->getRepository('AppBuncle:
             ContactoCuenta');
7           $contactoPrincipal = $contacto_cuenta_repo->findOneBy(array(
8            'cuenta' => $contacto_cuenta->getCuenta(),
9               'principal' => true
10          ));
11          $contactoPrincipal->setPrincipal(false);
12          $em->persist($contactoPrincipal);
13      }
14      //si no es principal, miro los contactos que tiene y si no hay, lo
            pongo como principal
15       else{
16           $contacto_cuenta_repo = $em->getRepository('AppBuncle:
                 ContactoCuenta');
17           $contactos = $contacto_cuenta_repo->findBy(
18               array('cuenta' => $contacto_cuenta->getCuenta())
19           );
20           if(count($contactos)==0){
21               $contacto_cuenta->setPrincipal(true);
22           }
23       }
24       $em->persist($contacto_cuenta);
25       $em->flush();
26
27        return $this->redirectToRoute('cuenta_show');
28
29  }
30  ...
```

5.4.2. Pruebas

En la interfaz de ver una oportunidad, se ha comprobado que los datos fueran los almacenados en la base de datos. También se ha comprobado que se muestran todos los productos de una oportunidad. Por último, se ha probado a añadir diferentes productos en una oportunidad y se ha comprobado que los resultados eran los esperados.

En la tabla de las cuentas de clientes, se han llevado a cabo las siguientes comprobaciones:

- Al acceder como administrador global en la aplicación se muestran todas las cuentas de la aplicación.
- Al acceder como gestor de aplicación en la aplicación se muestran todas las cuentas de la aplicación.
- Al acceder como gestor de oficina en la aplicación se muestran todas las cuentas de su oficina. Esto se ha probado con diferentes usuarios de diferentes oficinas.
- El filtrado individual de cada columna funciona correctamente.
- El filtrado múltiple filtra correctamente las oportunidades.

- El botón de borrar filtros elimina los filtros y actualiza los datos de la tabla.
- La paginación de la tabla permite desplazarse por todas las oportunidades de la aplicación.
- El botón para ver muestra correctamente los datos de la cuenta seleccionada.

En varias cuentas, se ha probado lo siguiente:

- Se ha comprobado que se mostraran todos los contactos de una cuenta, marcando el contacto principal con un icono de un tic.
- Se ha comprobado que al añadir un contacto a una cuenta sin contactos, este se guarde como principal, tanto para un contacto que el usuario ha marcado como principal, como para un contacto que el usuario no lo ha marcado.
- Se ha comprobado que al añadir un contacto a una cuenta con algún contacto preexistente, este se guarde como principal en caso de haber seleccionado esta opción. Por otra parte, que no se guarde como principal si no se ha seleccionado esta opción. Además de que no se modifique el contacto principal.
- Se ha comprobado que se muestran todos los productos de una cuenta. Estos productos son los que una cuenta compra directamente y no están relacionados con ninguna oportunidad.
- Se ha comprobado que al añadir un nuevo producto directamente a una cuenta, se muestra en la tabla de los productos de una cuenta.
- Se ha comprobado que al añadir varios productos a una oportunidad no altere los productos directamente asociados a una cuenta, y no se muestren en la tabla de los productos de una cuenta.
- Al añadir un producto en una cuenta, se ha comprobado que al modificar el área de negocio de un producto, se actualiza el tipo de producto.

5.5. Quinto sprint

En el seguimiento del quinto sprint (apartado 3.4.5.1) se ha mostrado las historias de usuarios que se han implementado durante este sprint. En concreto, se ha implementado el módulo de contactos web y parte de la gestión de usuarios.

5.5.1. Detalles de implementación

En primer lugar, se ha desarrollado el módulo de contactos web. Se ha creado el controlador base, la tabla del bundle, el formulario y las plantillas para las vistas.

En la tabla del bundle, se ha configurado el campo de propietario para que se pueda seleccionar el propietario directamente desde la tabla. en el código 5.9 se puede ver con más detalle.

En caso de no tener un propietario aparece el texto para añadir un propietario. Se puede ver que hay indicada una url en la que se va a enviar una petición. Por lo tanto, ha sido necesario crear un método en el controlador de contacto web para gestionar esta petición.

Código 5.9: Parte del código de la tabla de contacto web en la que se habilita la edición del propietario

```
1  ...
2  ->add('propietario.email', Column::class, array(
3   'default_content' =>  $this->translator->trans('TR_ADD'),
4   'title' => $this->translator->trans('TR_OWNER'),
5   'orderable' => false,
6   'searchable' => true,
7   'editable' => array(SelectEditable::class, array(
8       'source' => $options['usuarios'],
9    'mode' => 'inline',
10      'empty_text' => $this->translator->trans('TR_ADD'),
11   'url' => 'datatables_contacto_web_edit',
12   ))
13 ))
14 ...
```

El método que gestiona esta petición modifica el propietarios del contacto web y además, envía un correo al nuevo propietario, informándole de que se le ha asignado un contacto web. Por lo tanto, se ha tenido que configurar el Swiftmailer de Symfony para enviar este correo. Esto es indicar básicamente indicar la cuenta de correo y la contraseña.

En el código 5.10 se puede observar detalladamente el código que lleva a cabo este cometido. El controlador lee la clave primaria del contacto web a modificar de la petición y ,después, modifica el propietario entre las líneas 13-23. A continuación, en las líneas 26-33 se prepara y envía el correo al nuevo propietario.

Código 5.10: Método del controlador de contacto web que modifica el propietario y envía un correo informativo

```
1  /**
2  * Edit field.
3  * @param Request $request
4  * @Route("/sg/datatables/edit/contacto_web/field", name="
       datatables_contacto_web_edit")
5  * @Method("POST")
6  * @return Response
7  */
8  public function editDatatableAction(Request $request)
9  {
10  if ($request->isXmlHttpRequest()) {
11      // x-editable sends some default parameters
12      $pk = $request->request->get('pk');
13      $new_value = $request->request->get('value');
14
15   $em = $this->getDoctrine()->getManager();
16         $contacto_web_repo = $em->getRepository('AppBundle:ContactoWeb'
               );
17   $contacto_web = $contacto_web_repo->find($pk);
```

```
18          $usuario_repo = $em->getRepository('AppBundle:User');
19          $usuario = $usuario_repo->find($new_value);
20          $contacto_web->setPropietario($usuario);
21          // save all
22          $em->persist($contacto_web);
23          $em->flush();
24
25          $user= $this->get('security.token_storage')->getToken()->
                getUser();
26          $message = \Swift_Message::newInstance()
27              ->setSubject($this->get('translator')->trans('
                    TR_NEW_ASIGNED_BUDGET').': '.$contacto_web->getNombre()
                    .' | '.$contacto_web->getAsunto())
28              ->setFrom('info@cuatroochenta.com')
29              ->setTo($usuario->getEmail())
30              ->setBody($this->get('translator')->trans('TR_EMAIL_MESSAGE
                    ').$user->getUsername())
31          ;
32          $mailer = $this->get('mailer');
33          $result = $mailer->send($message);
34          return new Response($result, 200);
35      }
36      return new Response('Bad request', 400);
37  }
```

Por último, se ha pasado a realizar la gestión de los usuarios. Para ello, se ha utilizado el bundle FOSUserBundle. Se ha creado el controlador con las acciones de registro, edición y listado de usuarios. Estos métodos los proporciona el bundle, y se ha modificado principalmente las redirecciones que hace, de esta forma al registrar y editar usuarios se redirecciona a la tabla de los usuarios. Ya que es el administrador global el único usuario que puede crear y editar usuarios.

Para estas dos acciones, se han creado las plantillas para las vistas con el diseño proporcionado por Cuatroochenta. Estas plantillas se han insertado en el directorio *app\Resources\FOSUserBundle* para sobreescribir las vistas del bundle.

5.5.2. Pruebas

En la tabla de los contactos web, se han llevado a cabo las siguientes comprobaciones:

- Al acceder como administrador global en la aplicación se muestran todos los contactos web de la aplicación.
- Al acceder como gestor de aplicación en la aplicación se muestran todos los contactos web de la aplicación.
- Al acceder como gestor de oficina en la aplicación se muestran todos las contactos web de su oficina. Esto se ha probado con diferentes usuarios de diferentes oficinas.
- El filtrado individual de cada columna funciona correctamente.

- El filtrado múltiple filtra correctamente las oportunidades.
- El botón de borrar filtros elimina los filtros y actualiza los datos de la tabla.
- La paginación de la tabla permite desplazarse por todas las oportunidades de la aplicación.
- El botón para ver muestra correctamente los datos de la cuenta seleccionada.
- Se puede modificar el propietario de un contacto web que no tuviera propietario con anterioridad. Además, se envía el correo informando al nuevo propietario.
- Se puede modificar el propietario de un contacto web que ya tenía un propietario anteriormente. Además, se envía el correo informando al nuevo propietario.

En cuanto a la gestión de los usuarios, se ha comprobado registrar un nuevo usuario con los tres roles. Posteriormente, se comprueba que se puede acceder con este usuario creado, y que se pueden llevar a cabo las funcionalidades implementadas para cada tipo de usuario.

5.6. Sexto sprint

En el seguimiento del sexto sprint (apartado 3.4.6.1) se ha mostrado las historias de usuarios que se han implementado durante este sprint. Concretemente, se ha terminado de implementar la edición de usuarios. Además, se ha probado el funcionamiento completo de la aplicación. Por último, se ha publicado la aplicación en un servidor de Amazon, que han preparado en Cuatroochenta para albergar provisionalmente la aplicación.

5.6.1. Detalles de implementación

En primer lugar, se ha modificado el método de edición de usuarios. Este método se ha dividido en cuatro:

- Editar contraseña contraseña del usuario que está autenticado en la aplicación.
- Editar datos de perfil de un del usuario que está autenticado en la aplicación.
- Resetear la contraseña de cualquier usuario por parte del administrador global.
- Editar los datos de cualquier usuario por parte del administrador global.

Las dos primeras acciones son accesibles desde el menú superior de la aplicación, ya que están relacionadas con el usuario que esta autenticado en la aplicación. Se han hecho algunas modificaciones al código del bundle que permite realizar estas acciones. Principalmente, para controlar las redirecciones después de llevar a cabo cada tarea.

Las otras dos se ejecutan desde la tabla del bundle, que solo será accesible para el administrador global. En el código 5.11 se puede observar la configuración sobre el fichero de seguridad

de symfony, que controla el acceso de la aplicación. La línea 2 permite acceder a cualquier usuario al formulario de login. En la línea 3, se requiere del rol de administrador global para acceder al módulo de configuración, donde está la gestión de usuarios, oficina, moneda, etc. Por último, en la línea 4 se requiere que el usuario esté autenticado paraacceder a cualquier ruta de la aplicación.

Código 5.11: Parte del código del fichero de segurida, de la configuración de Symfony, que controla el acceso a la aplicación

```
1 access_control:
2         - { path: ^/[^/]+/login$, role: IS_AUTHENTICATED_ANONYMOUSLY }
3         - { path: ^/[^/]+/admin/, role: ROLE_ADMIN }
4         - { path: ^/[^/], role: IS_AUTHENTICATED_FULLY }
```

En el código anterior se puede observar que se controla el acceso a los directorios del segundo nivel. Esto sucede porque, en un primer nivel, va el lenguaje con el que se quiere acceder a la aplicación. *es* para el español y *en* para el inglés.

Después de esto, se ha dado por concluido el desarrollo de la aplicación.

Por último, se ha subido la aplicación de Symfony al servidor que ha preparado la empresa para albergar la aplicación. Seguidamente, se ha creado la base de datos en el nuevo servidor a partir del dump de la base de datos que se tenía en local.

5.6.2. Pruebas

En primer lugar, se ha comprobado que la edición de usuarios fuera la correcta. Para comprobar esta funcionalidad se ha hecho:

- Comprobar que como administrador se puede editar los datos de un usuario. Esto es posible para usuarios con el rol de administrador global, gestor de aplicación y gestor de oficina.
- Comprobar que como administrador puede resetear la contraseña de cualquier usuario. Además, el usuario editado puede volver a acceder a la aplicación con la nueva contraseña.
- Comprobar que como administrador se puede modificar la contraseña de su usuario. Después de esto, este usuario puede volver a autenticarse en la aplicación con la nueva contraseña.
- Comprobar que como administrador puede editar los datos de su propio usuario.
- Comprobar que como gestor de aplicación puede editar los datos de su propio usuario.
- Comprobar que como gestor de aplicación puede editar su contraseña. Después, se comprueba que puede volver a autenticarse en la aplicación.
- Comprobar que como gestor de oficina puede editar los datos de su propio usuario.
- Comprobar que como gestor de oficina puede editar su contraseña. Después se comprueba que puede volver a autenticarse en la aplicación.

Por último, se ha hecho una prueba general de la aplicación, en la que se ha probado la aplicación por completo y se ha visto que el funcionamiento era el esperado.

Capítulo 6

Conclusiones

Durante la estancia en prácticas, he asumido el reto de llevar a cabo el desarrollo de la aplicación CRM 480 en un framework desconocido como era Symfony. Además de esto, presentaba la dificultad añadida de que durante el Grado, apenas utilizamos PHP y por consiguiente, también era desconocido para mí. Durante la asignatura de Diseño e Implementación de Software había estado trabajando en un framework de Spring de Java para el desarrollo de aplicaciones web, que además utilizaba el patrón MVC. Esto me ha ayudado para poder aprender este nuevo framework, ya que al seguir el mismo patrón, algunos conceptos son iguales. Adicionalmente, durante los primeros días, hice un curso formativo en Symfony para familiarizarme con su funcionamiento.

Por otra parte, he conocido el funcionamiento de la empresa Cuatroochenta. Me ha gustado mucho el buen ambiente que se respira en la oficina. Esto te contagia a la hora de desempeñar tus labores y ayuda a integrarte con los compañeros de la oficina. Además los compañeros están dispuestos a ayudarte si te surge alguna duda. Aunque sea una empresa joven, afrontan con seriedad y buen hacer los proyectos que llevan a cabo. Además del trabajo de planificación que desempeñan, que tiene un papel fundamental en el éxito de los proyectos.

Por oro lado, a nivel personal, me ha servido para confiar en mis habilidades a la hora de enfrentarme a un desafío de este tipo, en el que conocía la tecnología que se iba a emplear. Además de conseguir formarme en Symfony. Aunque todavía me queda mucho por aprender. También ha sido la primera toma de contacto con el mundo laboral y la experiencia ha sido gratificante.

Por último, los resultados del proyecto, desde mi punto de vista, han sido satisfactorios. Se han conseguido la mayoría de los objetivos planteados al inicio de esta memoria. En cambio, algunos no se han podido desarrollar. Concretamente, el objetivo que no se ha podido llevar a cabo ha sido el de integrar la aplicación de presupuestos de Cuatroochenta. Este objetivo no se pudo cumplir ya que al disponer de poco tiempo, se priorizó otros aspectos más interesantes del CRM. Otro aspecto que no ayudó en el desarrollo del proyecto fue el de no disponer desde un primer momento de los diseños de la aplicación, ya que se tuvo que adaptar todas las plantillas al diseño, y es un esfuerzo que se hubiera podido evitar si hubieran estado desde un primer momento. Por último, la falta de algunas relaciones en el mapeado de la base de datos

repercutió negativamente ya que se tuvieron que revisar y añadir. Este problema estaba presente en las entidades de la aplicación de CRM que habían sido proporcionadas por la empresa. Estas adversidades han de servir para que en futuros proyectos no se tenga en cuenta desde el principio y no se vuelvan a repetir.

Bibliografía

[1] Amazon EC2. https://aws.amazon.com/es/ec2/. [Consulta: 22 de Febrero de 2018].

[2] BazingJsTranslationBundle. https://github.com/willdurand/BazingaJsTranslationBundle/. [Consulta: 3 de Febrero de 2018].

[3] BitBucket. https://es.atlassian.com/software/bitbucket. [Consulta: 26 de Febrero de 2018].

[4] Bootstrap. https://getbootstrap.com/. [Consulta: 26 de Febrero de 2018].

[5] COMPOSER. Dependency Manager for PHP. https://getcomposer.org/. [Consulta: 26 de Febrero de 2018].

[6] Configurar el inicio de sesión único (SSO) en cuentas gestionadas de Google utilizando proveedores de identidades de terceros. https://support.google.com/a/answer/60224?hl=es. [Consulta: 26 de Febrero de 2018].

[7] CRM y sus aplicaciones empresariales", author = "Macarena Reina. http://mundoerp.com/blog/crm-aplicaciones-empresariales/. [Consulta: 2 de Febrero de 2018].

[8] Descubre qué es un CRM y todas las ventajas que puede ofrecer a tu empresa. https://elcosmonauta.es/crm-customer-relationship-management/. [Consulta: 22 de Febrero de 2018].

[9] Doctrine. http://www.doctrine-project.org/. [Consulta: 26 de Febrero de 2018].

[10] FOSUserBundle. https://github.com/FriendsOfSymfony/FOSUserBundle. [Consulta: 3 de Febrero de 2018].

[11] HeidiSQL. https://www.heidisql.com/. [Consulta: 26 de Febrero de 2018].

[12] Jira Software. https://es.atlassian.com/software/jira. [Consulta: 20 de Febrero de 2018].

[13] jQuery. https://jquery.com/. [Consulta: 26 de Febrero de 2018].

[14] JSON. https://www.json.org/. [Consulta: 26 de Febrero de 2018].

[15] Material Design. https://material.io/. [Consulta: 26 de Febrero de 2018].

[16] Material Design for Bootstrap. https://mdbootstrap.com/. [Consulta: 26 de Febrero de 2018].

[17] MySQL. https://www.mysql.com/. [Consulta: 26 de Febrero de 2018].

[18] phpMyAdmin. https://www.phpmyadmin.net/. [Consulta: 26 de Febrero de 2018].

[19] PhpStorm. https://www.jetbrains.com/phpstorm/. [Consulta: 26 de Febrero de 2018].

[20] SgDatatablesBundle. https://github.com/stwe/DatatablesBundle/. [Consulta: 3 de Febrero de 2018].

[21] SourceTree. https://es.atlassian.com/software/sourcetree. [Consulta: 26 de Febrero de 2018].

[22] Symfony. https://symfony.com/. [Consulta: 26 de Febrero de 2018].

[23] The Scrum Guide. https://www.scrumguides.org/scrum-guide.html. [Consulta: 3 de Febrero de 2018].

[24] Twig. https://twig.symfony.com/. [Consulta: 26 de Febrero de 2018].

[25] WampServer. http://www.wampserver.com/en/. [Consulta: 26 de Febrero de 2018].

[26] Rodriguez Daniel. Cómo funciona un CRM y por qué permite vender más y mejor. https://www.gestiopolis.com/como-funciona-un-crm-y-por-que-permite-vender-mas-y-mejor/. [Consulta: 25 de Febrero de 2018].

[27] Bolsa de Valencia. Entorno Pre Mercado. http://www.bolsavalencia.es/esp/BValencia/ServEmpresas/Pre_Mercado_1.aspx. [Consulta: 2 de Febrero de 2018].

[28] Javier Eguiluz. *Desarrollo web ágil con Symfony2*. Easybook 4.9, 2017. p. 19, 20, 37.

[29] Soluciones Cuatroochenta S.L. Contacto web Soluciones Cuatroochenta. http://www.cuatroochenta.com/contacto/. [Consulta: 25 de Febrero de 2018].

[30] Soluciones Cuatroochenta S.L. Página web Soluciones Cuatroochenta. http://www.cuatroochenta.com. [Consulta: 2 de Febrero de 2018].