Aplicación SaaS para la gestión de pequeñas empresas.

Caso de aplicación: "Centro de Belleza"

José Luis Barrera Trancoso

ETIS

Oriol Martí Girona

Junio de 2014

Agradecimientos

A Mayra, por acompañarme todos estos años de duro esfuerzo y sacrificio, por su paciencia y comprensión, sin su apoyo no hubiera sido posible. A mi familia, por confiar en mí. A mi compañero de fatigas por su inestimable ayuda.

Resumen

Este documento es la memoria del Trabajo Final de Carrera (TFC) de Ingeniería del Software de Ingeniería Técnica en Informática de Sistemas. Los diferentes apartados de los que está compuesta detallan las diferentes fases del proyecto, desde la planificación inicial a su implementación.

El presente TFC se centra en el análisis y diseño de una aplicación web para la gestión de pequeñas empresas, con la particularidad de contar desde su inicio con un diseño que le permita:

- Poder ser desplegada en un entorno *cloud computing* y soportar múltiples clientes de forma concurrente (*multitenancy*)
- Poder adaptarse a cualquier tipo de negocio, permitiendo añadir nuevas funcionalidades o extender las existente en base a módulos

El objetivo del TFC no ha sido implementar una aplicación con funcionalidad completa, sino validar el análisis y el diseño realizados mediante la implementación concreta en una aplicación de gestión de centros de belleza.

Los **objetivos personales** de este proyecto han sido varios:

- Abordar, de forma simplificada, todas las fases por las que atraviesa un proyecto de desarrollo web, desde la planificación inicial, pasando por el análisis y diseño, hasta su implementación.
- Aprender nuevas tecnologías no utilizadas anteriormente.
- Sentar las bases para completar una aplicación de gestión para Centros de Bronceado para un familiar.

Palabras clave

TFC, SaaS, *multitenancy*, django, Python, git, github, centro belleza, Ingeniería del software, eclipse, bootstrap, jQuery.

Área

Ingeniería del software

Índice

Índice de ilustraciones

Capítulo 1 : Introducción

Justificación y contexto en el que se desarrolla

Este proyecto se enmarca dentro de la asignatura TFC, el objetivo de la cual es desarrollar (análisis, diseño e implementación) una solución tecnológica.

El proyecto escogido es una aplicación web para la gestión de un Centro de Belleza. El motivo de haber escogido este tipo de negocio y no otro, es debido al hecho de que un familiar es propietario de uno. Esto facilita enormemente el conocimiento del negocio, la obtención de la información necesaria y la identificación de necesidades concretas para llevar a cabo el desarrollo del mismo. Además, los resultados de este proyecto serán de utilidad directa en la vida real.

Se plantea el diseño y desarrollo de una aplicación web, capaz de desplegarse en un entorno *cloud computing*, y **proporcionarse como SaaS** (*Software as a Service*[1]), por lo que se tendrán en cuenta características propias de este tipo de aplicaciones para la su implementación (fundamentalmente el *multitenancy*[2]).

Como se ha mencionado anteriormente, **el diseño de la solución se concebirá en base a módulos**, de forma que sea posible extender el software a otro tipo de negocios, así como activar o desactivar módulos en función de las necesidades.

Objetivos

El presente proyecto tiene como objetivo principal el **desarrollo de una solución tecnológica que cubra las necesidades de gestión de una pequeña empresa**. En concreto se plantea como escenario un Centro de Belleza, por lo que la solución estará enfocada en dicho negocio y en sus necesidades específicas.

No obstante, otro de los objetivos que se persiguen conseguir durante el desarrollo del presente proyecto, es que el sistema a diseñar sea **modular y extensible**, de forma que pueda ser utilizado como base para futuros desarrollos de nuevas funcionales y extenderlo a otro tipo de negocios. Al mismo tiempo, se pretende utilizar una **arquitectura tecnológica moderna**, que permita que la aplicación pueda ser desplegada en un entorno *cloud computing* y diseñada para ofrecer un **modelo de explotación basado servicios**.

Enfoque y metodología

A lo largo del presente proyecto se pretende realizar el **análisis y diseño de una aplicación web para la gestión de un centro de belleza**. A su vez, se prevé realizar, a modo de validación, la **implementación** del modelo y de las funcionalidades básicas identificadas. A nivel de interfaz de usuario, se proporcionarán las pantallas o *templates* básicos para el manejo de la aplicación, dejando abierta la opción de incorporación de nuevos temas de diseño.

La aplicación se diseñará teniendo en mente la modularidad, la extensibilidad y el SaaS (*Software as a Service*) como características fundamentales de la misma.

[1] http://en.wikipedia.org/wiki/Software_as_a_service
[2] http://en.wikipedia.org/wiki/Multitenancy

En una **primera fase** se incluye la selección de la temática del proyecto y el desarrollo del plan de trabajo, donde se describen los objetivos que se pretenden alcanzar en el proyecto y se detalla la planificación a seguir durante su ejecución.

En la **segunda fase** se realizará el análisis y diseño del proyecto. Aquí se detallará la toma de requisitos, el análisis del sistema, explicando qué hace el software, y el diseño técnico para explicar cómo lo hace y qué tecnologías se utilizan.

La **tercera fase** incluye la implementación y las pruebas finales.

En la **última fase** se recopilan el código, todos los documentos realizados para crear la memoria final y la presentación del proyecto, completando de esta forma todos los entregables requeridos en el TFC.

Análisis de riesgos

Al ejecutarse el proyecto, las anteriores previsiones pueden variar por diversas causas, pues el proyecto no está exento de riesgos. Existen varios factores que pueden incidir directamente en el éxito del proyecto o reducir el grado de cumplimiento con los objetivos previstos. A continuación mencionamos algunos de los identificados:

Riesgo	Plan de contingencia
Desconocimiento de la tecnología	Se priorizarán aquellas tecnologías que dispongan de documentación y comunidad suficiente, y que proporcionen recursos, que permitan resolver cuestiones que surjan relacionadas con la tecnología.
Problemas de disponibilidad	Se revisará el plan de trabajo y la planificación de forma frecuente, con objeto de cumplir los plazos establecidos en la planificación inicial. Se establecerá un margen temporal de seguridad para cubrir posibles imprevistos.
Planificación deficiente	Se realizará un seguimiento técnico continuo de forma que se puedan detectar las posibles desviaciones a tiempo, con el fin de que puedan ser corregidas o mitigadas.
Problemas en la elección de las herramientas	Se realizará un estudio previo de las tecnologías más adecuadas para el proyecto, de forma que se reduzcan los riesgos por una mala elección tecnológica.

Planificación del proyecto

A continuación se describen brevemente las fases y tareas identificadas en el proyecto, así como la planificación de las mismas mediante un cronograma:

FASE I: Plan de trabajo

Esta fase se compone de las siguientes tareas:

- Decisión de la temática del proyecto
- Descripción del proyecto
- Identificación de las tareas
- Realización de la planificación del proyecto, incluyendo los hitos o entregas parciales (PACs) y el cronograma de tareas.

- Redacción del plan de trabajo

Listado de entregables:

- Plan de trabajo (PAC 1)

FASE II: Análisis y diseño

Esta fase se compone de las siguientes tareas:

- Identificación de los requisitos formales a los que deberá darse solución en el proyecto
- Definición del alcance del proyecto, en base a los requisitos seleccionados
- Descripción de los casos de uso
- Modelización en base a prototipos de los elementos y disposición de la interfaz gráfica
- Modelización de las clases de objetos y relaciones
- Diseño de la arquitectura tecnológica que dará soporte a los desarrollos
- Identificación de las tecnologías para la implementación del modelo y la configuración de las herramientas que se utilizarán en el entorno de desarrollo
- Documentación del modelo de la base de datos

Listado de entregables:

- Documento de análisis y diseño (PAC 2)

FASE III: Implementación

Esta fase se compone de las siguientes tareas:

- Instalación y configuración del entorno de desarrollo
- Implantación y despliegue inicial de una aplicación referencia basada en la arquitectura seleccionada
- Implementación del modelo de clases
- Implementación de la lógica de negocio (funcionalidades de la aplicación)
- Diseño básico de las plantillas de la interfaz de usuario

Listado de entregables:

- Código fuente (PAC 3)

FASE IV: Publicación de resultados

Esta fase se compone de las siguientes tareas:

- Composición y revisión de la memoria final del proyecto
- Preparación de la presentación
- Pruebas y corrección de errores en la aplicación

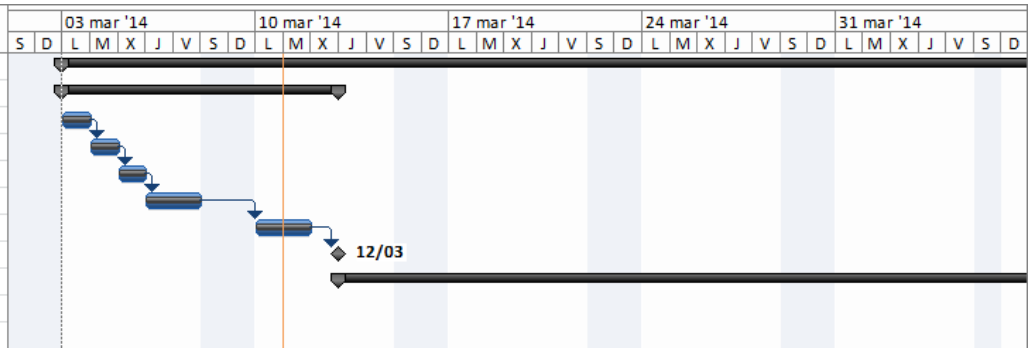
Listado de entregables:

- Memoria del proyecto
- Presentación del proyecto

Cronograma

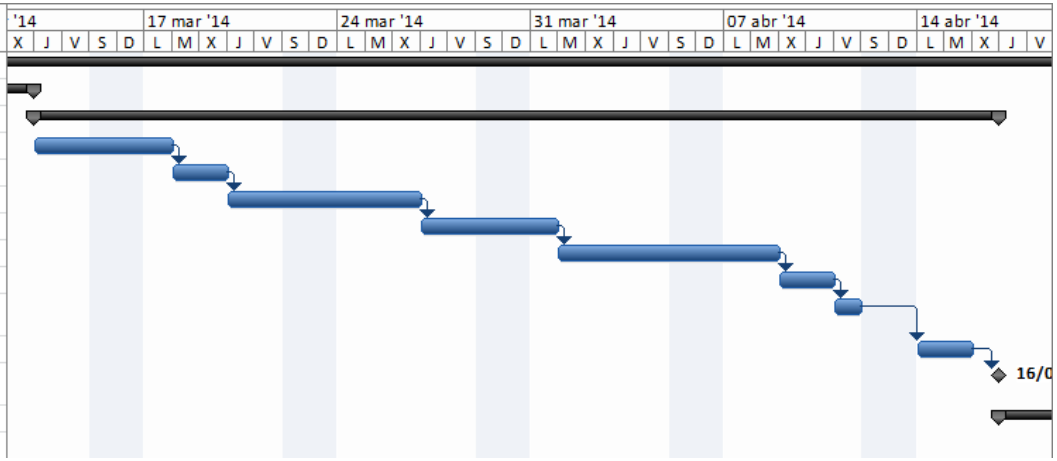
FASE I:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**TFC**	**73 días**	**lun 03/03/14**	**mié 11/06/14**
2	**FASE I: Plan de trabajo**	8 días	lun 03/03/14	mié 12/03/14
3	Decisión de la temática del proyecto	1 día	lun 03/03/14	lun 03/03/14
4	Descripción del proyecto	1 día	mar 04/03/14	mar 04/03/14
5	Identificación de tareas	1 día	mié 05/03/14	mié 05/03/14
6	Planificación del proyecto	2 días	jue 06/03/14	vie 07/03/14
7	Redacción Plan de trabajo	2 días	lun 10/03/14	mar 11/03/14
8	Entrega del Plan de trabajo (PAC 1)	1 día	mié 12/03/14	mié 12/03/14
9	**FASE II: Análisis y diseño**	**25 días**	**jue 13/03/14**	**mié 16/04/14**
19	**FASE III: Implementación**	**25 días**	**jue 17/04/14**	**mié 21/05/14**
26	**FASE IV: Publicación de resultados**	**15 días**	**jue 22/05/14**	**mié 11/06/14**



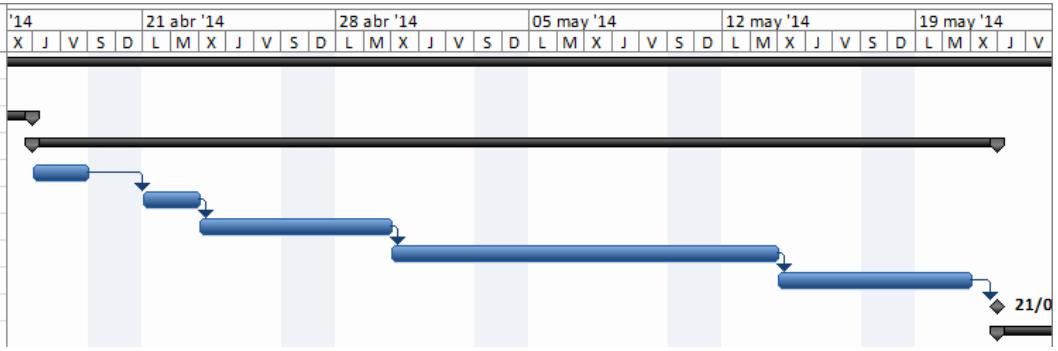
FASE II:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**TFC**	**73 días**	**lun 03/03/14**	**mié 11/06/14**
2	**FASE I: Plan de trabajo**	8 días	lun 03/03/14	mié 12/03/14
9	**FASE II: Análisis y diseño**	**25 días**	**jue 13/03/14**	**mié 16/04/14**
10	Obtención de requisitos	3 días	jue 13/03/14	lun 17/03/14
11	Definición del alcance del proyecto	2 días	mar 18/03/14	mié 19/03/14
12	Casos de uso	5 días	jue 20/03/14	mié 26/03/14
13	Modelización de la Interfaz Gráfica	3 días	jue 27/03/14	lun 31/03/14
14	Modelización de clases de objetos y relaciones	6 días	mar 01/04/14	mar 08/04/14
15	Diseño de la arquitectura tecnológica	2 días	mié 09/04/14	jue 10/04/14
16	Identificación de tecnologías para la implementación y desarrollo	1 día	vie 11/04/14	vie 11/04/14
17	Diseño del modelo de la base de datos	2 días	lun 14/04/14	mar 15/04/14
18	Entrega del Documento de Análisis y Diseño (PAC 2)	1 día	mié 16/04/14	mié 16/04/14
19	**FASE III: Implementación**	**25 días**	**jue 17/04/14**	**mié 21/05/14**
26	**FASE IV: Publicación de resultados**	**15 días**	**jue 22/05/14**	**mié 11/06/14**



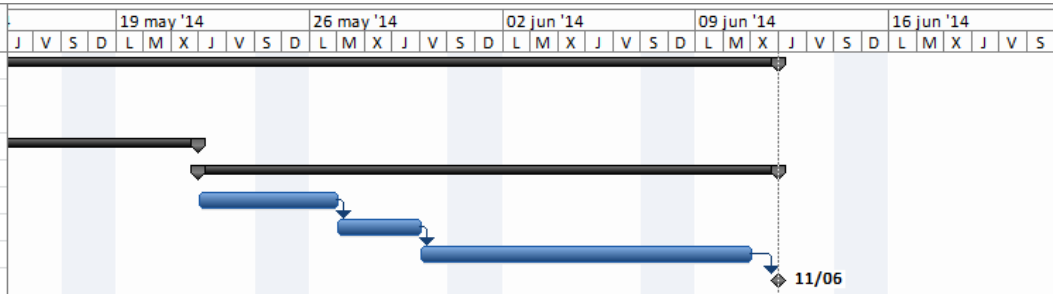
FASE III:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**TFC**	**73 días**	**lun 03/03/14**	**mié 11/06/14**
2	**FASE I: Plan de trabajo**	8 días	lun 03/03/14	mié 12/03/14
9	**FASE II: Análisis y diseño**	**25 días**	**jue 13/03/14**	**mié 16/04/14**
19	**FASE III: Implementación**	**25 días**	**jue 17/04/14**	**mié 21/05/14**
20	Configuración del entorno de desarrollo	2 días	jue 17/04/14	vie 18/04/14
21	Implantación inicial de la arquitectura	2 días	lun 21/04/14	mar 22/04/14
22	Implementación del modelo	5 días	mié 23/04/14	mar 29/04/14
23	Implementación de la lógica de negocio	10 días	mié 30/04/14	mar 13/05/14
24	Diseño básico de plantillas	5 días	mié 14/05/14	mar 20/05/14
25	Entrega del código fuente (PAC 3)	1 día	mié 21/05/14	mié 21/05/14
26	**FASE IV: Publicación de resultados**	**15 días**	**jue 22/05/14**	**mié 11/06/14**



FASE IV:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**TFC**	**73 días**	**lun 03/03/14**	**mié 11/06/14**
2	**FASE I: Plan de trabajo**	8 días	lun 03/03/14	mié 12/03/14
9	**FASE II: Análisis y diseño**	**25 días**	**jue 13/03/14**	**mié 16/04/14**
19	**FASE III: Implementación**	**25 días**	**jue 17/04/14**	**mié 21/05/14**
26	**FASE IV: Publicación de resultados**	**15 días**	**jue 22/05/14**	**mié 11/06/14**
27	Revisión de la memoria final del proyecto	3 días	jue 22/05/14	lun 26/05/14
28	Preparación de la presentación	3 días	mar 27/05/14	jue 29/05/14
29	Pruebas y corrección de errores	8 días	vie 30/05/14	mar 10/06/14
30	Entrega Memoria y Presentación	1 día	mié 11/06/14	mié 11/06/14



Productos obtenidos

Como resultado del TFC se han obtenido los siguientes productos:

- Memoria del Trabajo Final de Carrera.
- Presentación del Trabajo Final de Carrera
- Código Fuente de la aplicación web desarrollada

Breve descripción del resto de capítulos

En los próximos capítulos de la memoria, se detallan el resto de fases contempladas en la planificación, como son el análisis funcional y el diseño técnico, así como la implementación llevada a cabo a partir de éstos.

Del mismo modo, se detalla la valoración económica del proyecto, y se exponen las conclusiones personales extraídas del presente TFC.

Capítulo 2 : Análisis Funcional

En este capítulo se realiza el análisis teórico del sistema, incluyendo los requisitos detectados, los casos de uso concretos, el alcance, así como los prototipos de interfaces de usuario.

Breve descripción de la aplicación

El objetivo de este análisis está centrado en una aplicación web de gestión para centros de belleza, orientada a gestionar el trabajo diario que se realiza en el centro. No obstante, el análisis se realiza desde la base que la aplicación tiene que ser lo suficientemente flexible para ser extendida en funcionalidades y a otro tipo de negocios. Así como que pueda ser utilizada por varios clientes al mismo tiempo, es decir, que varios centros de belleza independientes puedan utilizar la misma instalación de la aplicación y la misma base de datos, en un servidor en la nube, trabajando exclusivamente con los datos que le son de su propiedad y sin posibilidad de acceder a datos que no le pertenezcan.

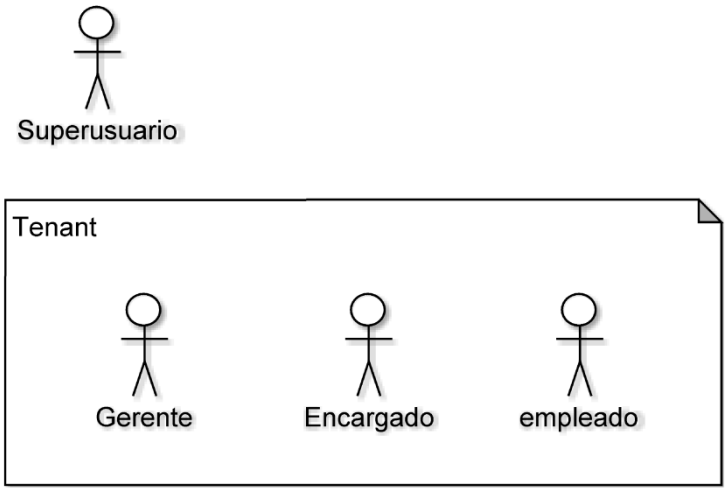
Actores

En una aplicación de gestión como la que nos ocupa, el sistema debe estar diseñado para soportar diferentes usuarios con diferentes roles y niveles de permisos. Esto es así, ya que no todos los empleados deben poder acceder a toda la información del sistema, ni manejar determinadas secciones del software.

En el caso concreto del centro de belleza, el usuario que da de alta el Centro (rol gerente) dispondrá de todos los permisos sobre sus contenidos. No obstante, se hace necesario en la práctica, que el centro de belleza se gestione en el día a día con un único usuario con permisos controlados (rol empleado). Así mismo, podrá existir un usuario con permisos extendidos (rol encargado) con capacidad de gestionar más entidades y asumir mayor responsabilidad.

También debe existir un súper-usuario administrador del cloud o de la aplicación en su globalidad, con capacidad de gestionar toda la aplicación y todos los tenants. Este usuario no pertenece a ningún tenant, sino al dueño del SaaS.

A continuación se muestran los actores implicados en la aplicación:



- **Superusuario**. Es el dueño de la infraestructura y dispone de todos los permisos sobre los contenidos del SaaS, incluyendo la de todos los tenants.
- **Tenant**. Es un cliente de la aplicación de gestión. No es un rol como tal, sino que un tenant hace referencia a los datos de un cliente, es decir, de una empresa o centro de belleza. Un tenant dispone como mínimo de un usuario con rol gerente, no obstante, podrá disponer de tantos usuarios con otros perfiles como estime oportuno:
    - **Gerente**. Dispone de todos los privilegios sobre los datos de su negocio en la aplicación. Tiene todos los permisos para gestionar todas las entidades definidas: Empleados, Clientes, Ingresos, Gastos, Empresa, Productos, Servicios, Proveedores, Bonos y Citas.
    - **Encargado**. Es la persona en la empresa que dispone de privilegios para gestionar todas las entidades, excepto los datos de la empresa y de los empleados.
    - **Empleado**. Sólo tiene permisos para gestionar determinadas entidades: Ingresos, Gastos, Clientes, Bonos, Citas.

Requisitos funcionales

A continuación se detallan los requisitos identificados en una primera fase, y que han sido considerados como como punto de partida básico, para cubrir las necesidades de funcionamiento de un negocio de este tipo:

RF-001	
Descripción	Los empleados de un Centro de Belleza deben poder identificarse en el sistema mediante usuario y contraseña

RF-002	
Descripción	El sistema permitirá gestionar (crear, editar y eliminar) las diferentes entidades utilizadas en la aplicación: empleados, clientes, servicios, productos, proveedores, bonos, citas, ventas y gastos.

RF-003	
Descripción	Un empleado sólo debe poder ver los datos correspondientes al Centro de Belleza donde trabaje.

RF-004	
Descripción	El sistema debe permitir controlar el stock de productos.

RF-005	
Descripción	Los bonos están compuestos por una serie de sesiones de servicios que los clientes pueden utilizar. El sistema debe permitir asignar un bono a un cliente y controlar las sesiones que ha gastado, así como aquellas que le queden disponibles.

RF-006	
Descripción	El sistema debe controlar la caducidad de los bonos, en función de la duración establecida para cada uno de ellos.

RF-007	
Descripción	El sistema debe poder planificar una sesión de bono en un día y hora concreto.

RF-008	
Descripción	El sistema debe poder planificar las citas, incluyendo la fecha y hora, la duración estimada, los servicios a realizar y el cliente que realiza la cita.

RF-009	
Descripción	El sistema debe poder registrar cualquier movimiento económico, tanto de entrada, como de salida.

RF-010	
Descripción	El sistema debe poder registrar las ventas realizadas y controlar el pago de los servicios, bonos, citas y productos.

RF-011	
Descripción	El sistema debe poder diferenciar entre las diferentes formas de pago: efectivo, tarjeta, domiciliación o transferencia.

RF-012	
Descripción	El sistema debe permitir realizar el cierre de caja.

RF-013	
Descripción	El sistema debe permitir buscar por las diferentes entidades registradas en el sistema

Requisitos no funcionales

Con objeto de cubrir los objetivos técnicos marcados en el proyecto, de proporcionar una arquitectura tecnológica moderna y un modelo de explotación del software siguiendo las tendencias actuales, se han identificado los siguientes requisitos no funcionales:

RNF-001	
Descripción	El sistema debe estar desarrollado utilizando tecnologías web

RNF-002	
Descripción	El sistema debe tener un diseño modular, de forma que permita la ampliación de funcionalidades en base a módulos

RNF-003	
Descripción	El sistema debe ser *multitenancy*, es decir, debe estar diseñado de forma que soporte múltiples usuarios en una misma instancia y base de datos.

RNF-004	
Descripción	El sistema debe controlar la propiedad de los datos de cada tenant.

RNF-005	
Descripción	El sistema debe estar diseñado para permitir un modelo de explotación del software como servicio (SaaS)

RNF-006	
Descripción	El sistema debe estar diseñado para soportar varios idiomas

Alcance

Desarrollar una aplicación como la que nos ocupa requiere de un esfuerzo elevado en cuanto a la dedicación de recursos de desarrollo, más aún si consideramos abordar la totalidad de los requisitos identificados en el análisis.

Por este motivo, y en base a los requisitos identificados y a la disponibilidad real, se ha definido el siguiente alcance, respecto a la implementación del sistema con objeto de **validar el diseño de la solución** y sentar las bases para la ampliación del sistema:

- **Configuración del entorno de desarrollo**. Se llevará a cabo la puesta en marcha del entorno de desarrollo, incluyendo el repositorio de código y versionado, las herramientas de *debug*, el IDE y entornos de ejecución necesarios.
- **Configuración del el entorno de despliegue de la aplicación**. Se establecerá el entorno de despliegue de la aplicación, incluyendo la infraestructura, servidores de base de datos, etc.
- **Implementación de la arquitectura de referencia**. Se llevará a cabo el desarrollo de la arquitectura que dé soporte al resto de los módulos de la aplicación, incluyendo el *multitenancy*, la internacionalización, la modularidad, etc.
- **Implementación el núcleo de la aplicación**. La aplicación está compuesta por varios módulos, siendo el núcleo o core, la parte básica y compartida por todos. Se realizará:
    - Implementación del modelo.
    - Implementación de la lógica de negocio para las entidades principales.
    - Diseño básico de plantillas para la interfaz de usuario.
- **Implementación de un módulo externo**. De forma que extienda el núcleo del sistema proporcionando funcionalidades adicionales.

Casos de uso

CU-001	**Creación de usuario**
Descripción	Creación de un usuario en el sistema
Actor principal	Gerente de un Centro de Belleza
Precondiciones	El nombre de usuario no puede existir en la base de datos
Flujo de eventos	1. El usuario accede a la aplicación 2. Introduce los datos básicos para la creación de un usuario 3. El usuario es creado en el sistema
Flujo alternativo	
Flujo excepcional	2. El usuario ya existe en el sistema a. Muestra un mensaje indicando que el usuario ya existe en el sistema

CU-002	**Inicio de sesión**
Descripción	Inicio de sesión en el sistema
Actor principal	Cualquier usuario registrado
Precondiciones	El usuario debe existir previamente en el sistema
Flujo de eventos	1. El usuario accede a la aplicación 2. Introduce sus credenciales 3. El sistema valida los datos introducidos 4. Se inicia la sesión en la aplicación
Flujo alternativo	

Flujo excepcional	1. Introduce una credenciales erróneas a. Muestra un error indicando que el nombre de usuario o la contraseña introducidas no son correctas

CU-003	**Cambio de contraseña**
Descripción	Cambio de contraseña del usuario
Actor principal	Cualquier usuario del sistema
Precondiciones	El usuario debe haber iniciado sesión en la aplicación
Flujo de eventos	1. El usuario introduce su contraseña antigua y la contraseña nueva. Ésta última 2 veces 2. El sistema valida que la contraseña antigua es correcta 3. El sistema valida que la contraseña nueva coincide 4. El sistema cambia la contraseña del usuario
Flujo alternativo	
Flujo excepcional	1. El usuario introduce mal la contraseña antigua a. El sistema muestra un mensaje indicando que la contraseña no es correcta 1. La contraseña nueva escrita por el usuario dos veces no coincide a. El sistema muestra un mensaje indicando que ambas deben coincidir

CU-004	**Gestión de entidades (CRUD)**
Descripción	Gestión (CRUD) de las entidades gestionables en el sistema (empleados, clientes, servicios, productos, proveedores, bonos, citas, ventas y gastos)
Actor principal	Gerente, encargado o empleado (según permisos)
Precondiciones	El usuario debe tener permisos para gestionar la entidad
Flujo de eventos	1. El usuario accede a la entidad 2. El sistema muestra un listado de las entidades del sistema
Flujo alternativo	2. Crear una nueva entidad a. El usuario indica que quiere crear una nueva entidad b. El usuario introduce los datos necesarios c. El sistema valida la información introducida d. El sistema guarda la entidad en la base de datos 2. Borrar una entidad a. El usuario indica que quiere eliminar una entidad b. El sistema pide confirmación de la acción c. El sistema elimina la entidad 2. Modificar una entidad a. El usuario indica que quiere modifica una entidad b. El sistema muestra al usuario los datos de la entidad c. El usuario modifica los datos d. El sistema valida la información introducida e. El sistema guarda la entidad en la base de datos

Flujo excepcional	1. El usuario cancela la creación de una entidad a. El sistema no hace nada y vuelve al listado de entidades 2. El usuario cancela la edición de una entidad a. El sistema no hace nada y vuelve al listado de entidades 3. El usuario cancela la eliminación de una entidad a. El sistema no hace nada y vuelve al listado de entidades 4. El usuario introduce datos incorrectos a. El sistema muestra un error indicando que los datos introducidos no son correctos

CU-005	**Control del stock**
Descripción	Control del stock de productos en el Centro de Belleza
Actor principal	Cualquier usuario del sistema
Precondiciones	Debe existir el producto a consultar en la base de datos
Flujo de eventos	1. El usuario accede al producto que desea consultar su stock 2. El sistema muestra el stock disponible para ese producto
Flujo alternativo	1. El usuario realiza una venta de un producto a. El sistema modifica el stock del producto automáticamente 2. El usuario hace una compra de un producto a un proveedor a. El sistema modifica el stock del producto automáticamente
Flujo excepcional	3. El producto no está disponible en stock a. El sistema muestra un mensaje indicando que no hay existencias

CU-006	**Gestión de bonos y sesiones**
Descripción	Gestión de los bonos y de sus sesiones por parte de los usuarios
Actor principal	Cualquier usuario del sistema
Precondiciones	El bono debe haber sido creado previamente y asignado a un cliente
Flujo de eventos	1. El usuario accede al bono del cliente 2. El usuario descuenta una sesión al cliente 3. El sistema valida que el bono no está caducado 4. El sistema valida que dispone de sesiones disponibles 5. El sistema registra la sesión
Flujo alternativo	1. El usuario planifica una sesión del bono
Flujo excepcional	3. El sistema indica que el bono está caducado y no descuenta la sesión 4. El sistema indica que no hay sesiones disponibles y no descuenta la sesión

CU-007	**Control de caducidad de bonos**
Descripción	Control de la caducidad de los bonos
Actor principal	Cualquier usuario del sistema

Precondiciones	El bono debe estar creado en la base de datos
Flujo de eventos	1. El usuario accede al detalle del bono 2. El usuario comprueba la fecha de caducidad del bono
Flujo alternativo	2. El usuario quiere modificar la fecha de caducidad del bono a. Accede a la edición del bono b. Modifica la fecha de caducidad
Flujo excepcional	2. La fecha de caducidad introducida es incorrecta a. El sistema muestra un mensaje indicando que la fecha es incorrecta y no hace nada

CU-008	**Planificación de una sesión de un bono**
Descripción	Planificar una sesión de un bono
Actor principal	Cualquier usuario del sistema
Precondiciones	El bono debe existir y debe tener sesiones disponibles
Flujo de eventos	1. El usuario accede al bono del cliente 2. El usuario accede a la planificación de una sesión 3. El sistema le pide la fecha y la hora 4. El usuario planifica la sesión 5. El sistema registra la cita
Flujo alternativo	
Flujo excepcional	1. El usuario introduce los datos incorrectos o no completa algún dato obligatorio a. El sistema muestra un mensaje de error al usuario

CU-009	**Planificación de una cita**
Descripción	Planificar una cita a un cliente
Actor principal	Cualquier usuario del sistema
Precondiciones	1. El cliente debe existir en la base de datos 2. Los servicios a realizar en la cita deben existir en la base de datos
Flujo de eventos	1. El usuario accede a la opción para crear una cita 2. Introduce el cliente 3. Introduce los datos asociados a la cita (día, hora, duración, …) 4. Introduce los servicios a realizar 5. El sistema almacena la cita en la base de datos
Flujo alternativo	1. El cliente no existe en la base de datos a. El usuario da de alta al cliente en la base de datos 2. El servicio no existe en la base de datos a. El usuario da de alta el servicio en la base de datos
Flujo excepcional	2. El usuario introduce los datos incorrectos o no completa algún dato obligatorio a. El sistema muestra un mensaje de error al usuario

CU-010	**Ver calendario de citas**
Descripción	Mostrar el calendario de citas del día
Actor principal	Cualquier usuario del sistema

Precondiciones	
Flujo de eventos	1. El usuario accede al día que quiere visualizar en el calendario 2. El sistema muestra el listado de citas registradas para ese día
Flujo alternativo	

CU-011	**Realización de una venta**
Descripción	Realizar una venta (cobrar una cita o vender y producto)
Actor principal	Cualquier usuario del sistema
Precondiciones	1. Todas las entidades involucradas deben estar registradas en el sistema (cliente, servicios, productos, citas)
Flujo de eventos	1. El usuario indica que quiere cobrar la cita o el producto 2. El sistema crea una venta, calcula el precio y lo muestra al usuario 3. El usuario confirma los datos e introduce el método de pago y la cantidad entregada por el cliente 4. El sistema registra la venta como cobrada y le indica la vuelta (si el pago es en efectivo)
Flujo alternativo	3. El usuario modifica el precio de la venta a. El usuario confirma los datos e introduce el método de pago y la cantidad entregada por el cliente b. El sistema registra la venta como cobrada y le indica la vuelta (si el pago es en efectivo)

CU-012	**Registro de salida de efectivo**
Descripción	Registrar un movimiento de salida de efectivo en la caja
Actor principal	Cualquier usuario del sistema
Precondiciones	
Flujo de eventos	1. El usuario indica que quiere retirar efectivo de la caja 2. El usuario introduce la cantidad a retirar 3. El sistema registra la cantidad como un movimiento de salida de efectivo
Flujo alternativo	

CU-013	**Cierre de caja**
Descripción	Realizar el cierre de caja
Actor principal	Cualquier usuario del sistema
Precondiciones	
Flujo de eventos	1. El usuario indica que quiere hacer el cierre de caja 2. El sistema le muestra la cantidad que debería haber en la caja
Flujo alternativo	

CU-014	**Búsqueda de entidades**
Descripción	Buscar entidades en el sistema
Actor principal	Cualquier usuario del sistema
Precondiciones	

Flujo de eventos	1. El usuario indica que quiere realizar una búsqueda 2. El usuario introduce el texto a buscar 3. El sistema busca coincidencias en el nombre de las entidades 4. El sistema muestra al usuario las coincidencias encontradas
Flujo alternativo	1. El sistema no muestra ninguna coincidencia a. El usuario repite la búsqueda con otros parámetros

Diagrama de casos de uso

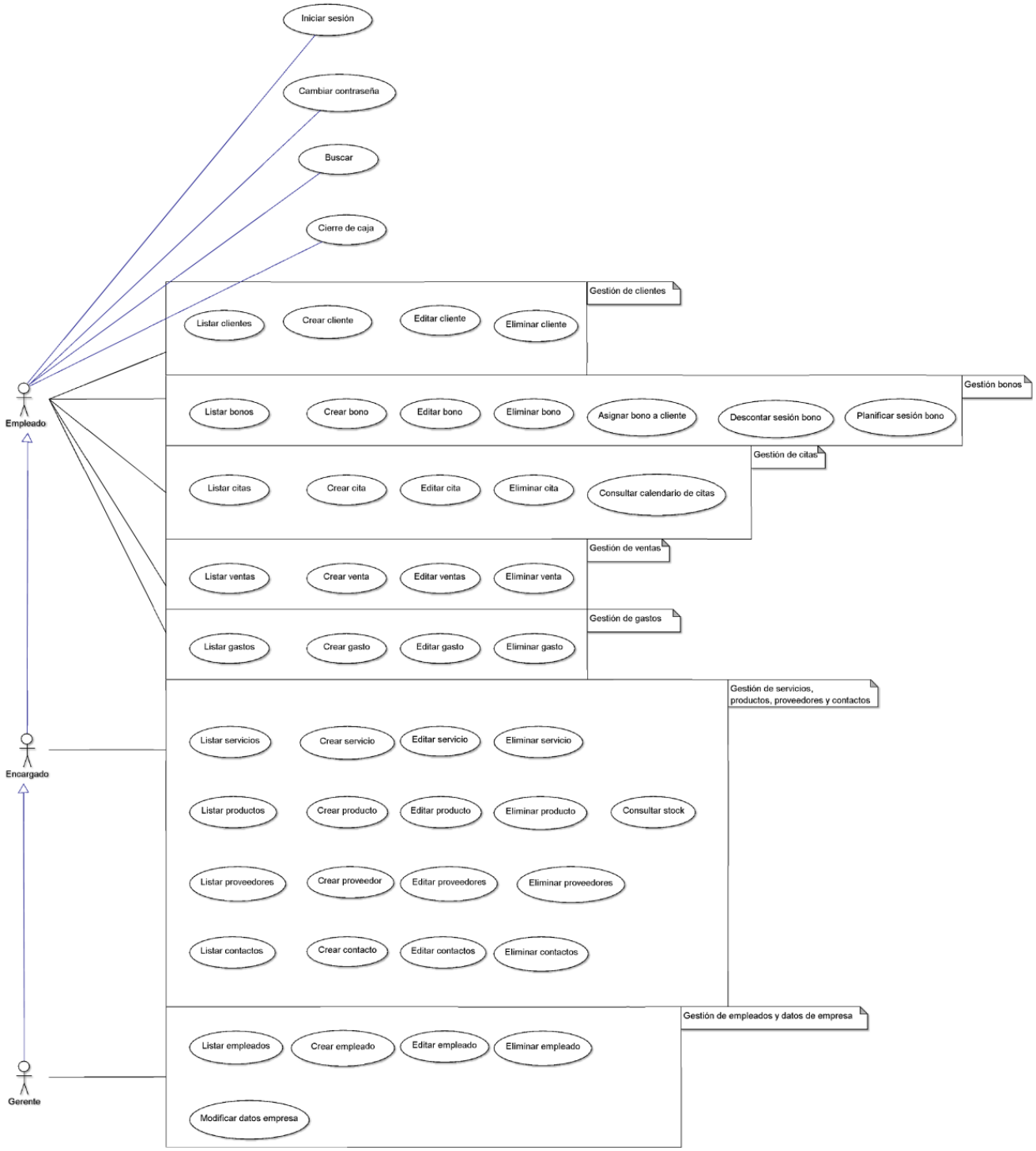


Ilustración 1: Diagrama de casos de uso

Diagrama de actividades

A continuación, y con objeto de clarificar los procesos de la herramienta, se muestra el flujo que sigue el proceso de creación de una cita, uno de los diagramas de actividades más complejos que han sido diseñados e implementados en la aplicación. En él podemos observar cómo se interrelacionan los diferentes módulos que forman parte del sistema.

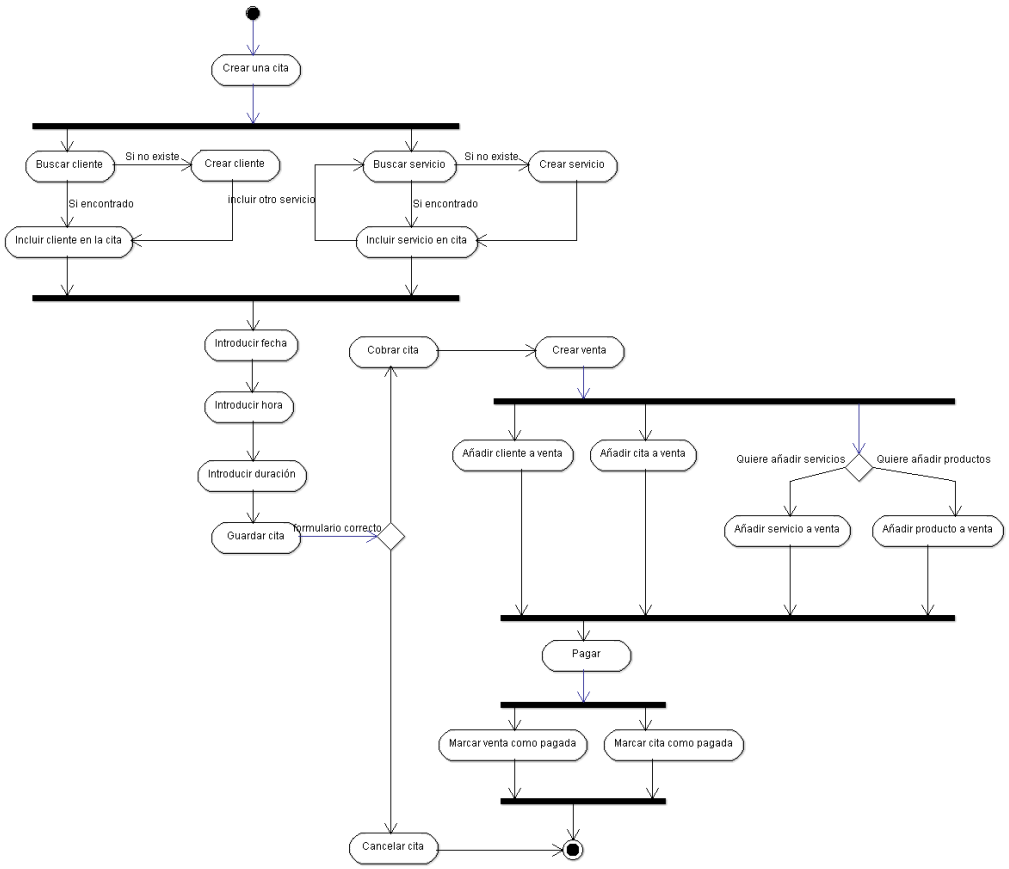


Ilustración 2: Diagrama de actividades creación de cita

Diagrama funcional

A continuación se muestra el diagrama de bloques funcionales que han sido incluidos en el análisis del sistema. Como se puede observar, está dividido en módulos que, a su vez, contienen una serie de sub-módulos.

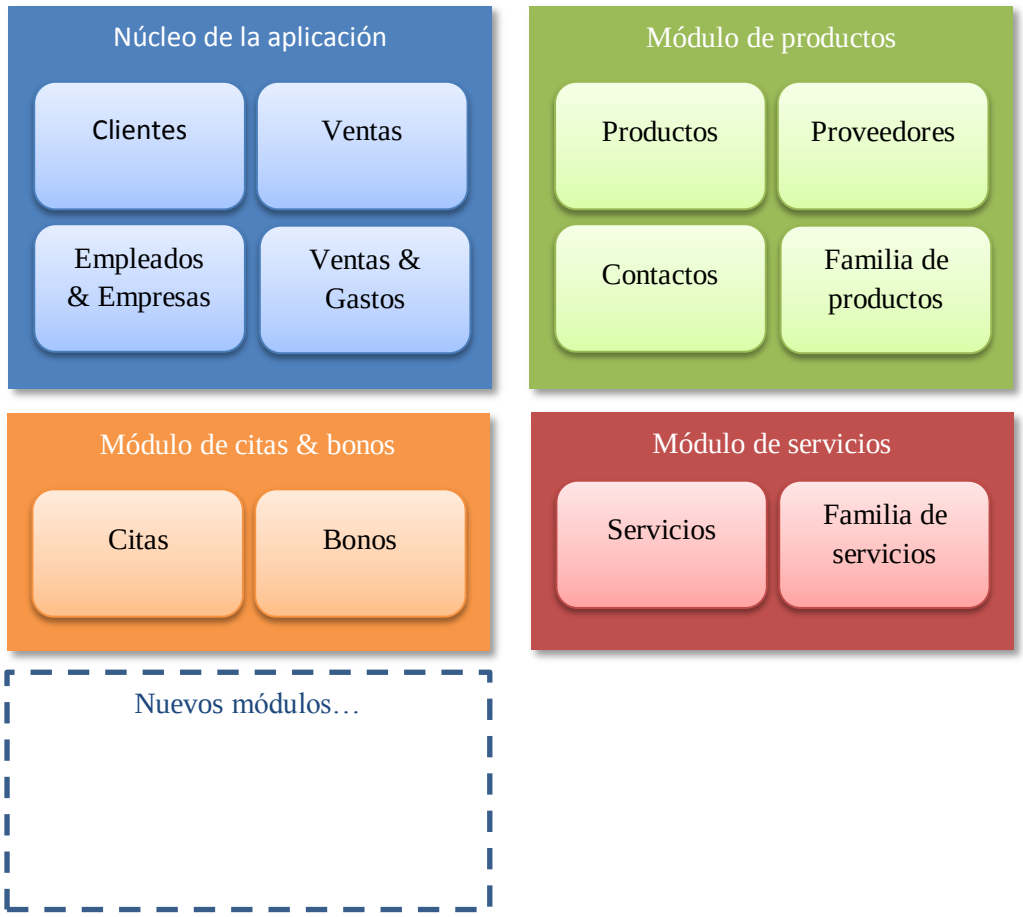


Ilustración 3: Diagrama funcional

- El **núcleo de la aplicación** está compuesto por las funcionalidades básicas de cualquier PYME, es decir: clientes, ventas, gastos y empleados.
- El **módulo de productos** contiene las funcionalidades básicas para la gestión de productos: Productos, proveedores, contactos y familia de productos.
- El **módulo de citas & bonos** proporciona las funcionalidades para la gestión de citas y de bonos por parte de los clientes, así como las sesiones de los bonos y la planificación de las mismas.
- El módulo de **servicios** permite gestionar los servicios ofrecidos y categorizarlos.

Cada uno de estos bloques funcionales podrá extenderse en un futuro si se considera necesario, únicamente añadiendo el sub-módulo correspondiente al módulo en cuestión. Además, podrán añadirse nuevos módulos al sistema con objeto de ampliar las funcionalidades del sistema, como podría ser: un módulo financiero, un módulo de facturación, un módulo de gestión de proyectos, un módulo de RRHH, un módulo de

tramitación, etc.

Prototipos de interfaces

Para el acceso a la aplicación, es necesario que los usuarios inicien sesión en el sistema, para ello se presentará una pantalla básica de *login* tal y como se puede observar en el siguiente prototipo:

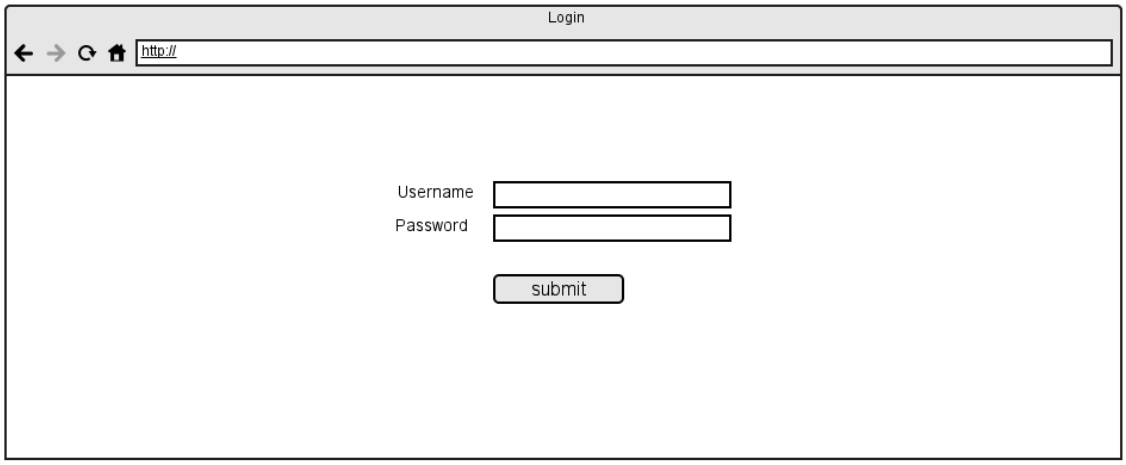


Ilustración 4: Interfaz usuario. Inicio de sesión

Del mismo modo, para el mantenimiento de las entidades definidas en el sistema, se diseñará una interfaz de usuario básica que permita realizar las operaciones CRUD (*Create, Read, Update y Delete*).

A continuación se puede ver un prototipo de interfaz de usuario para estas operaciones. En función del tipo de entidad, podrán cambiar los siguientes elementos:

- Listados: Selección de las columnas a mostrar en la tabla de listado en función de la entidad.
- Detalle: Muestra los parámetros específicos de cada entidad
- Edición: Muestra el formulario de edición con los campos definidos para la entidad
- Creación: Igual que la edición

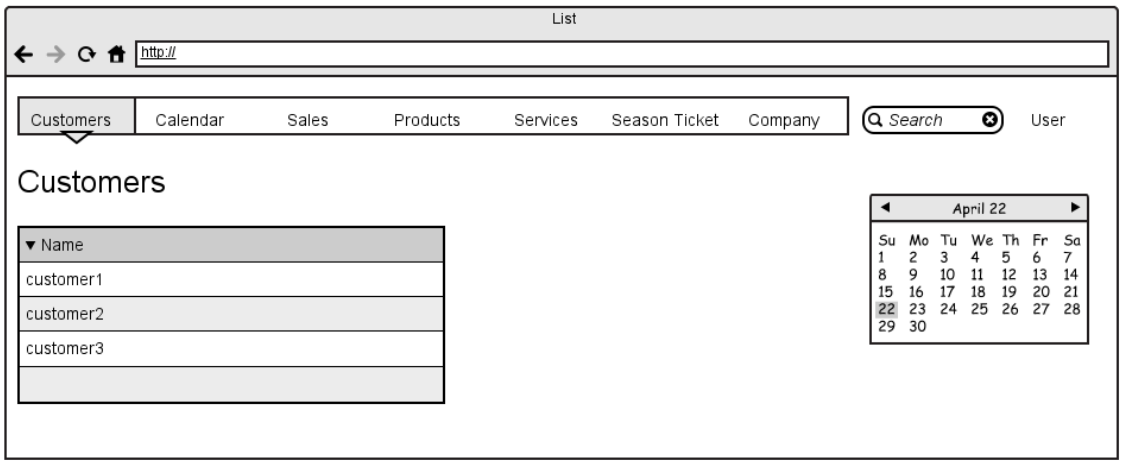


Ilustración 5: Interfaz de usuario. Listados

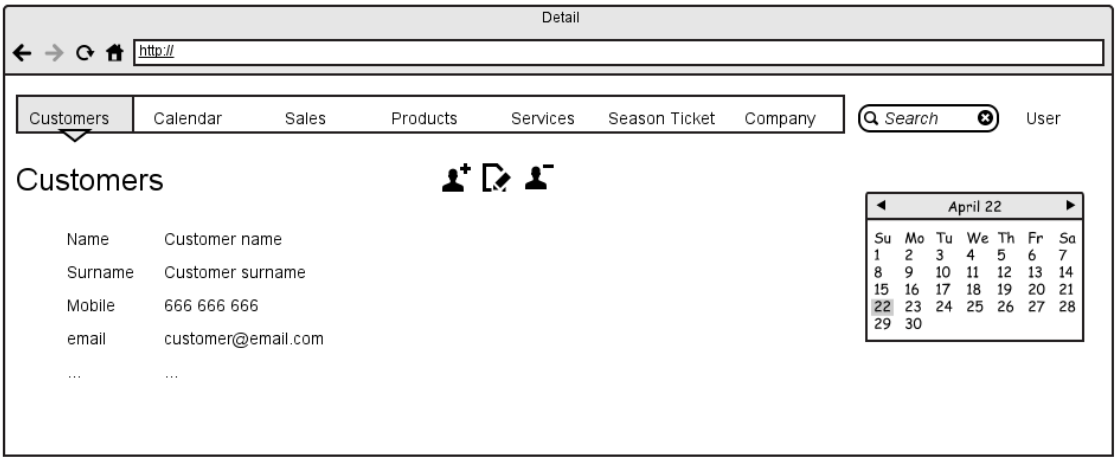


Ilustración 6: Interfaz de usuario. Detalle

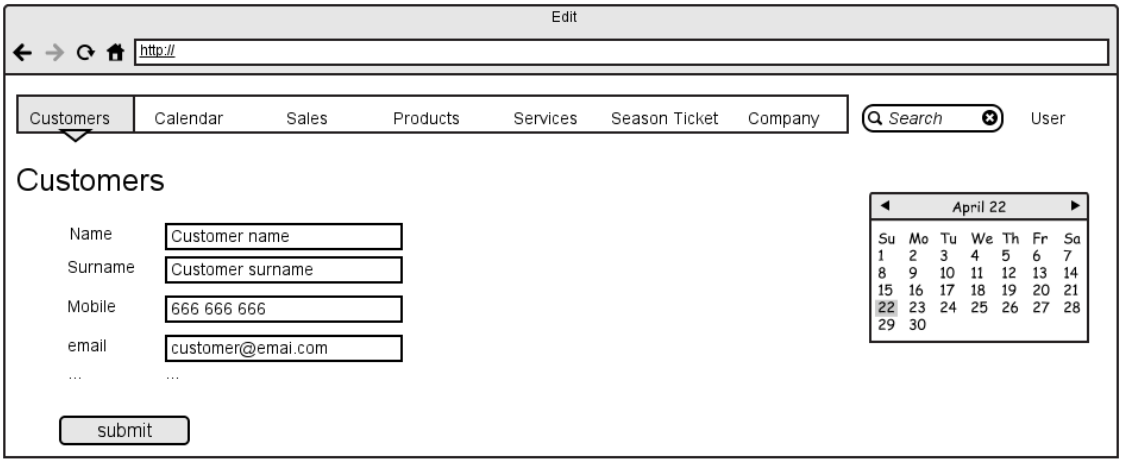


Ilustración 7: Interfaz de usuario. Edición

La aplicación dispondrá de una caja de búsqueda que será visible desde cualquier sección de la misma. De forma similar a los listados, los resultados de búsqueda se mostrarán en forma de lista tal y como se puede observar en el siguiente prototipo de pantalla:

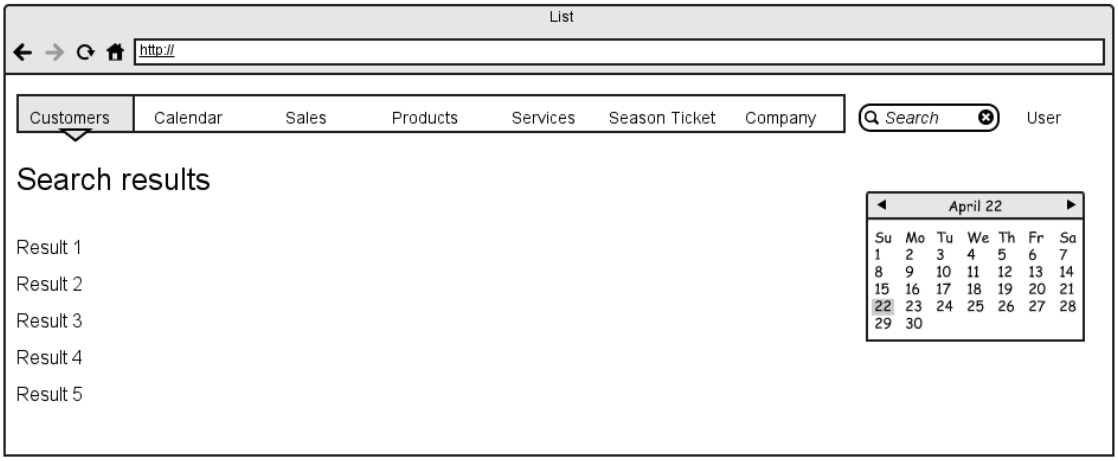


Ilustración 8: Interfaz usuario. Resultados de búsqueda

Además del modo listado, para el caso de las citas, se proporcionará una pantalla especial que permitirá identificar las citas de un día completo visualmente mediante un *timeline,* en el que cada cita será representada con un bloque, cuyo alto dependerá de la duración de la cita y que será mostrado sobre una rejilla horaria. A continuación se muestra un esbozo del interfaz:

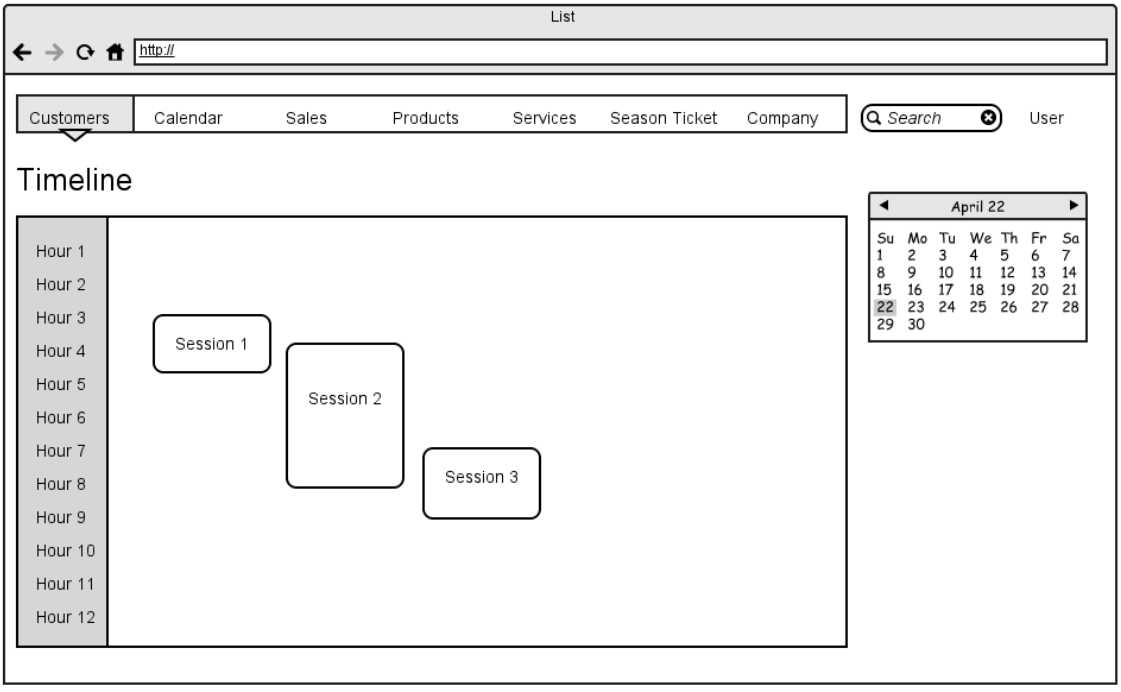


Ilustración 9: Interfaz usuario. Línea del tiempo

Capítulo 3 : Diseño técnico

Diagrama de clases

Como se ha explicado anteriormente, la aplicación dispone de un diseño modular, por ello, el único requisito para su funcionamiento básico es utilizar el núcleo o *core* del sistema, el uso del resto de módulos es opcional y dependerá de las necesidades específicas de cada pyme.

Para una mayor claridad en la comprensión del modelo, se exponen los diagramas de clases de forma modular. Finalmente, se muestra el diagrama del modelo completo con todos los módulos definidos en el análisis del sistema.

En primer lugar se muestra el diagrama de clases correspondiente al núcleo de la aplicación:

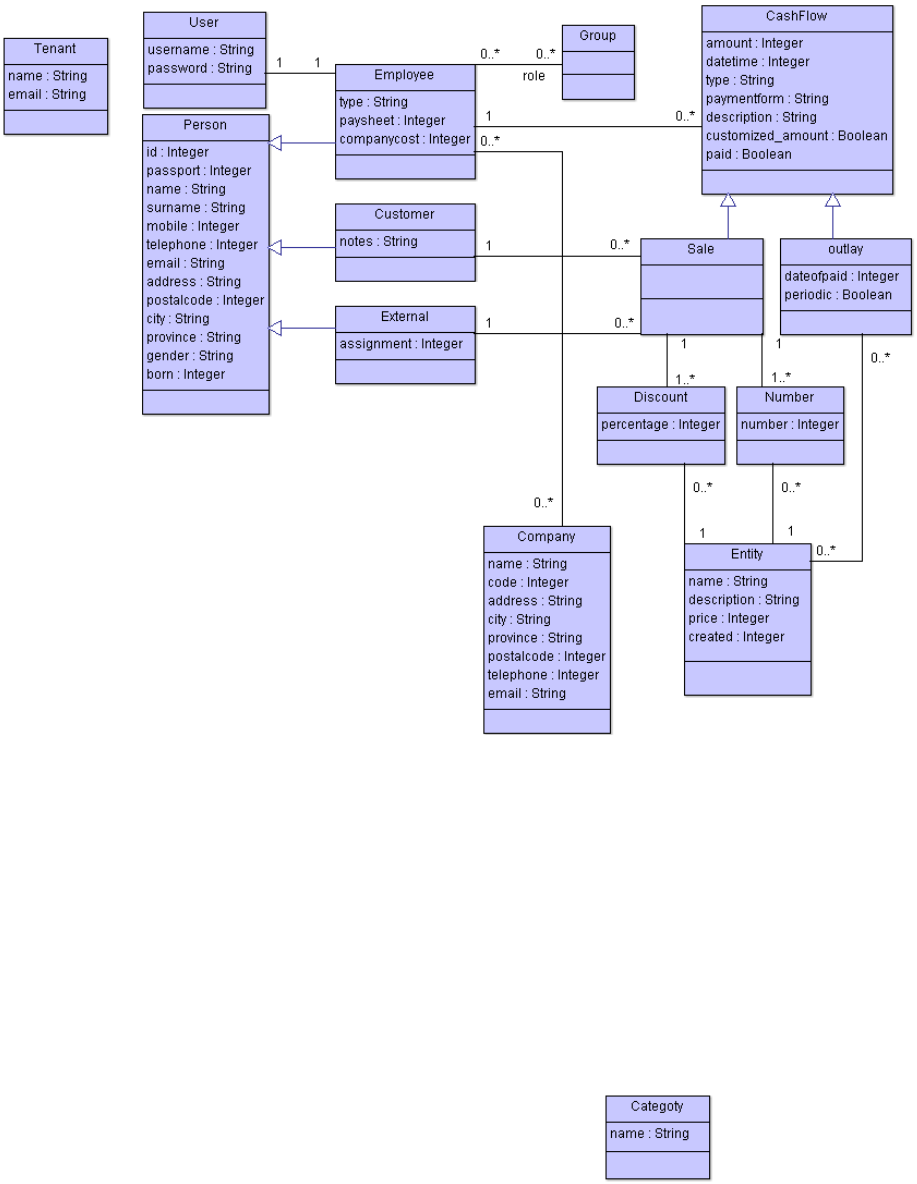


Ilustración 10: Diagrama de clases. Núcleo de la aplicación

La entidad Person modela un objeto de tipo persona y contiene los atributos básicos de cualquier persona que sea modelada en el sistema. Varias entidades heredan de Person como Employee, Customer y External. Este último es una entidad especial que modela colaboraciones externas de una persona que ofrece sus servicios y puede ir a comisión.

A su vez, la entidad Employee es la única que puede hacer login en el sistema mediante usuario y contraseña.

Las entidades Person, Company, Entity, CashFlow, Discount, Number y Category heredan de Tenant, la cual es la entidad encargada de gestionar la propiedad de los contenidos en la aplicación.

CashFlow es la entidad que gestiona las entradas o salidas de dinero, éstas pueden ser una venta (Sale), o cualquier tipo de gasto (outlay). A su vez, en una venta (Sale) pueden estar incluidas Entity (Servicios, Productos, Citas,…) y cada uno de los servicios incluidos en una venta pueden tener un número determinado de unidades, así como un descuento específico para esa venta.

Un Employee estará asociado a un CashFlow, esto implica que todo movimiento de caja, ya sea de entrada (una venta, ingreso efectivo, subvención,…) o de salida (pago proveedor, compras, impuestos,…) deberán estar asociados a un empleado que constará como responsable del movimiento.

El núcleo puede ser extendido añadiendo el módulo de productos tal y como se puede observar a continuación:

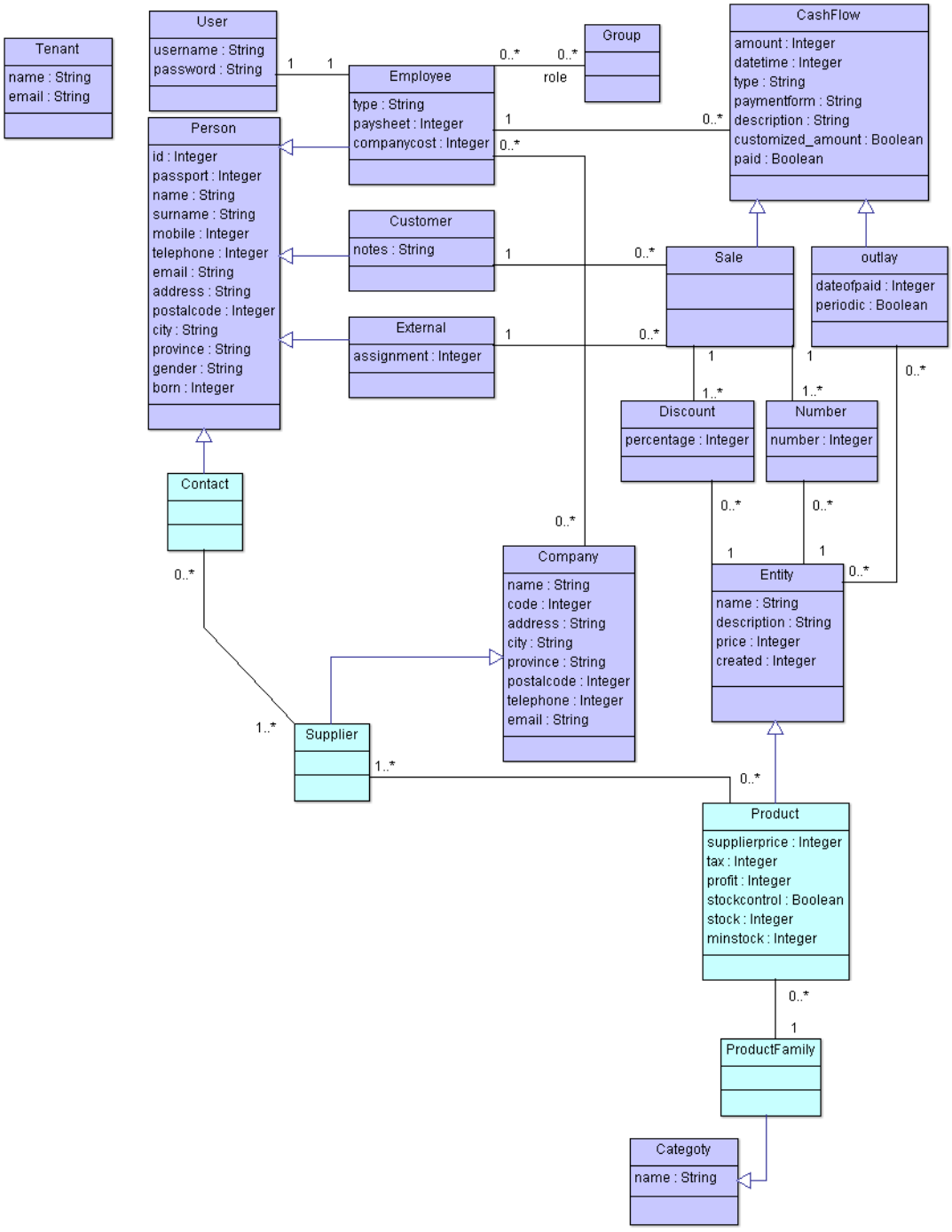


Ilustración 11: Diagrama de clases. Núcelo y Productos

Del mismo modo, el núcleo puede ser extendido añadiendo módulo de servicios:

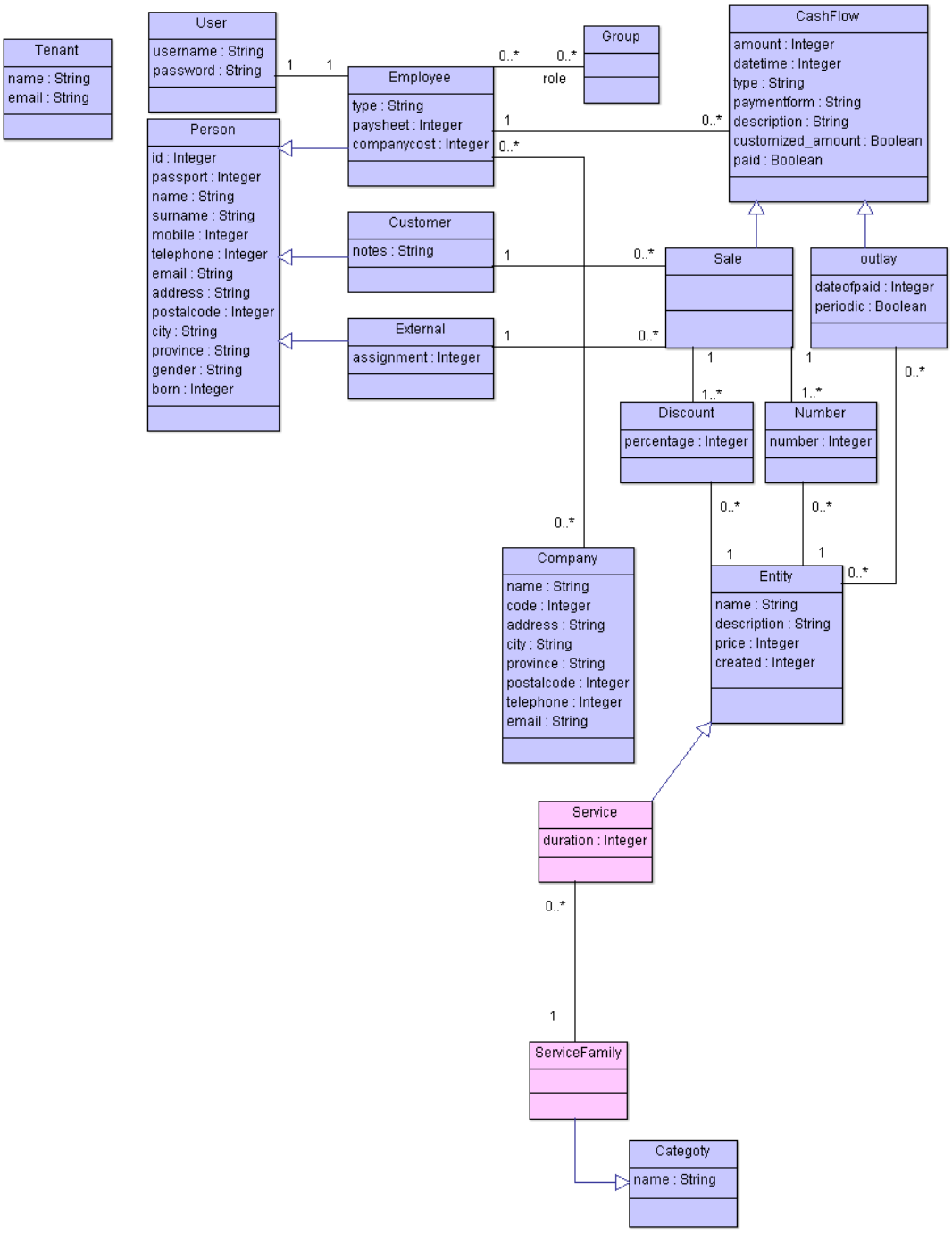


Ilustración 12: Diagrama de clases. Núcleo y Servicios

Y, del mismo modo, el núcleo puede ser extendido añadiendo el módulo de citas:

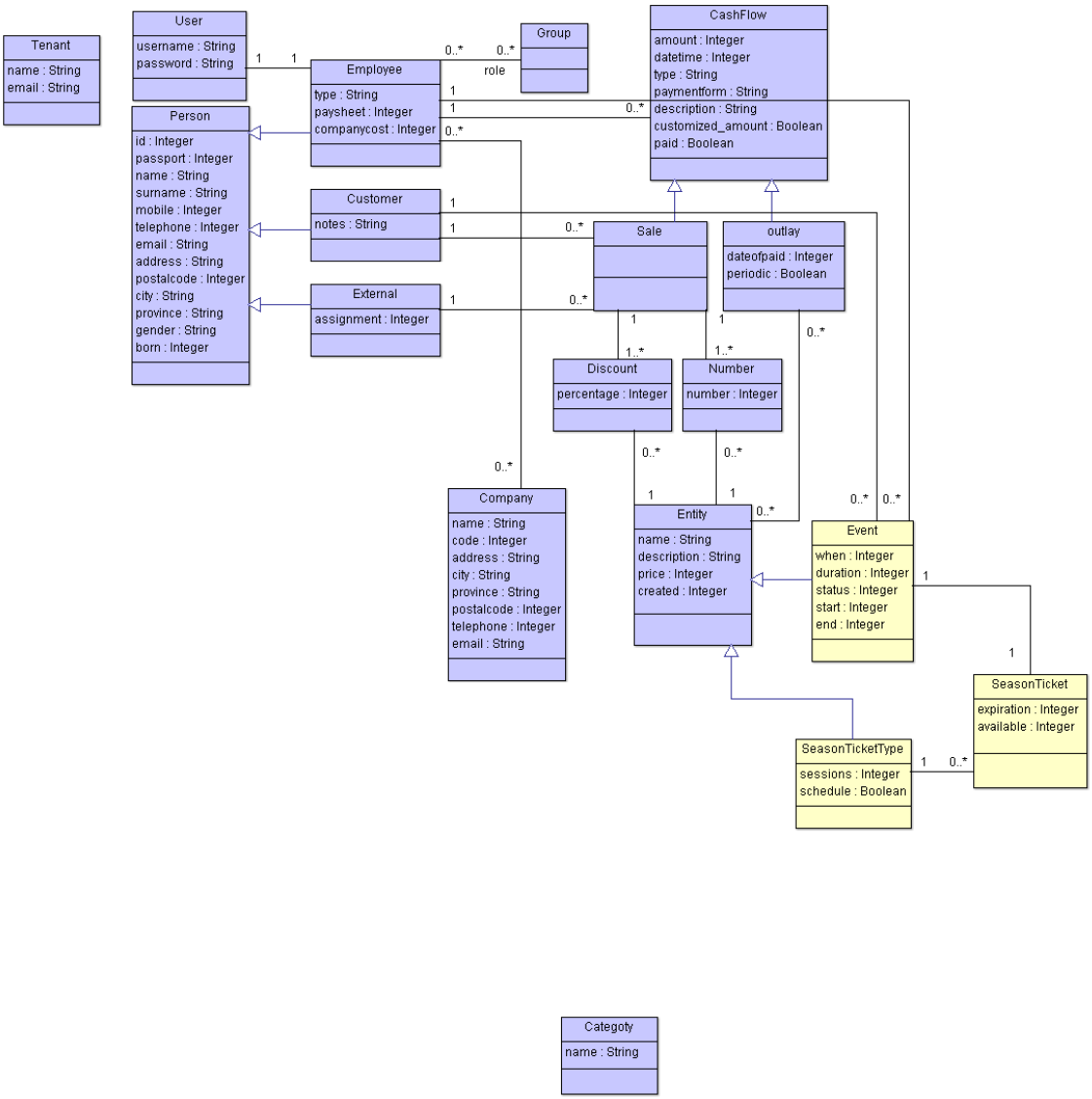


Ilustración 13: Diagrama de clases. Núcleo, bonos y citas

Finalmente, el diagrama completo de la aplicación con todos los módulos analizados sería el siguiente:

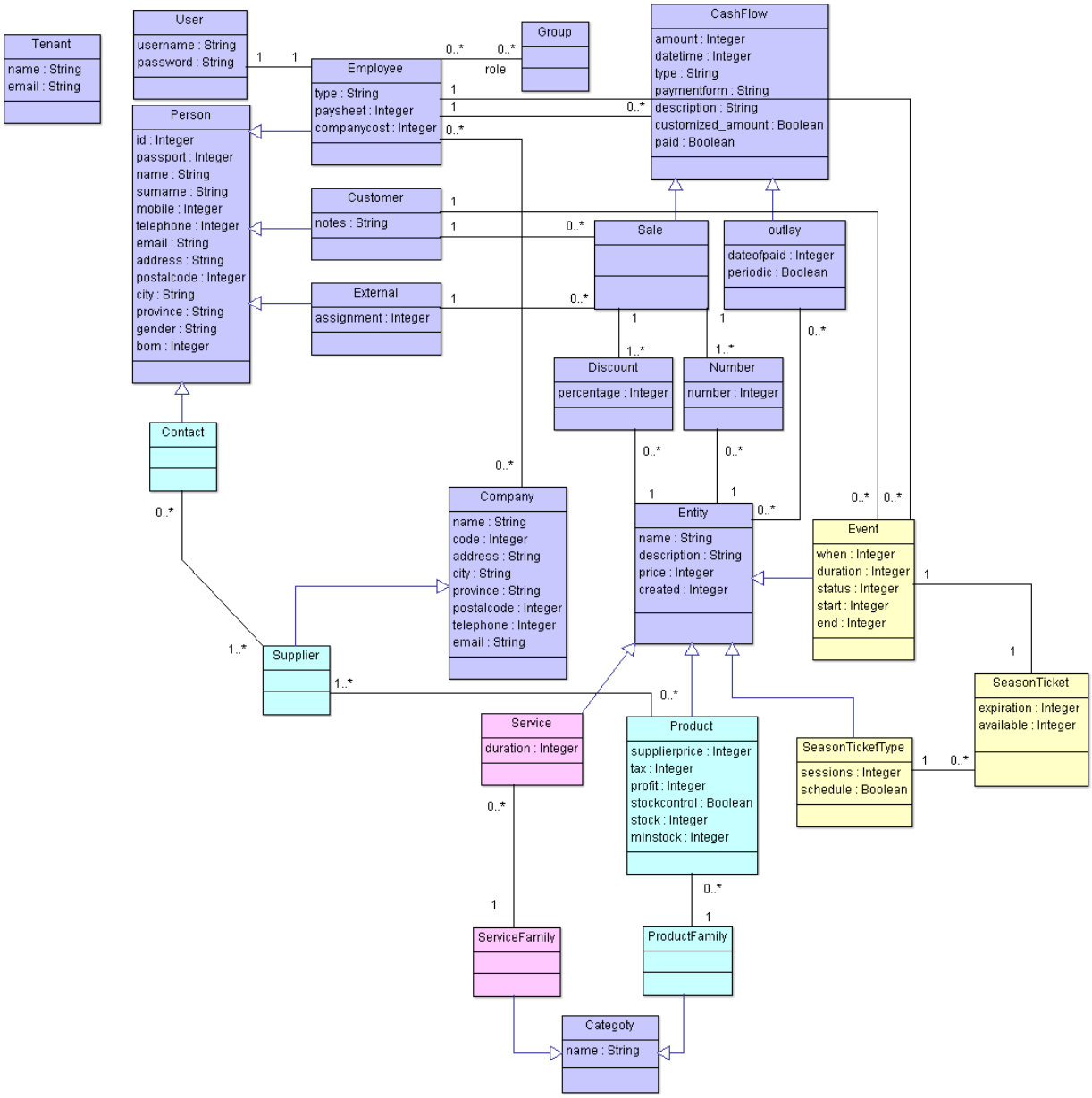


Ilustración 14: Diagrama de clases. Aplicación completa

Como se puede observar, los módulos no tienen dependencias entre sí, sólo con el núcleo de la aplicación, no obstante, podría darse el caso de que se desarrollaran módulos que dependan de otros módulos. Esto sería viable, sin embargo implicaría tener documentados los requisitos y versionados los módulos, ya que las evoluciones y actualizaciones de unos, podrán afectar a otros.

Arquitectura tecnológica

Después de un estudio de las tecnologías disponibles con las que se podría llevar a cabo el desarrollo de la aplicación y, con objeto de facilitar la creación de la aplicación siguiendo el

modelo y requisitos definidos, se ha decidido utilizar el framework web **django**[3].

Django es un framework *open source*, que sigue el paradigma MTV (Model, Template and View). Está escrito en Python[4] y puede ser utilizado para el desarrollo de aplicaciones web. Se alinea perfectamente a los objetivos del presente proyecto, ya que django pone especial énfasis en la reutilización y la extensibilidad de componentes.

La arquitectura de django y sobre la que se implementará la aplicación será la siguiente:

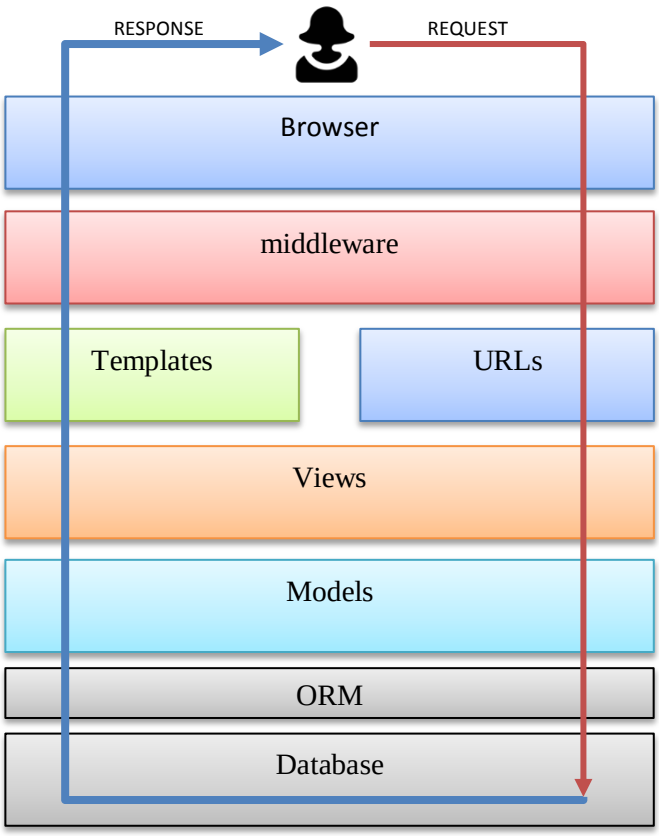


Ilustración 15: Arquitectura de django

Algunas de las características del framework son las siguientes:

- Implementa un ORM (mapeador objeto-relacional)
- Un API para la base de datos
- Capacidad de extensión mediante la incorporación de nuevas aplicaciones
- Sistema de vistas genéricas
- Sistema de plantillas basado en etiquetas
- Dispone de un creador de URL amigables basado en expresiones regulares
- Dispone de un middleware para implementar funcionales adicionales
- Internacionalización

[3] https://www.djangoproject.com/
[4] https://www.python.org/

El resto de componentes que se utilizarán para el desarrollo de la aplicación han sido los siguientes:

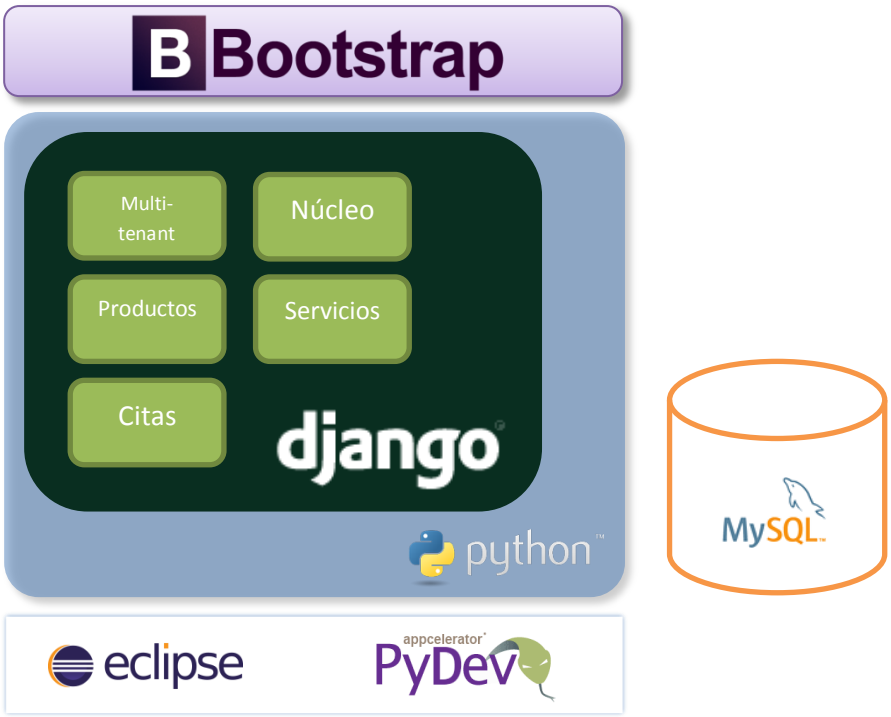


Ilustración 16: Arquitectura técnica del sistema

- **Django**, se utilizará la versión 1.6 del framework
- La base de datos será **MySQL**, no obstante django soporta varias bases de datos, por lo que podrían utilizarse otras (PostgreSQL, SQLite, Oracle).
- ***Multitenancy***. Se hará uso de la aplicación open source desarrollada por Philippe Ombredanne[5]: django-simple-multitenant[6], para implementar la tenencia múltiple en la aplicación, por lo que se adaptarán los modelos y vistas para que trabajen con ella.
- Se hará uso de la capacidad de django para crear e "instalar" aplicaciones a una aplicación existente, para cubrir la **modularidad** de la aplicación. Cada módulo funcional identificado, será implementado como una aplicación django. Los nuevos desarrollos serán incluidos como aplicaciones.
- El entorno de desarrollo utilizado será **eclipse** junto con el **plugin pyDev.**
- Intérprete de Python: **Python 2.7**
- Librerías de Python para la conexión con la base de datos MySQL: **MySQL-python-1.2.4**

Software adicional:

- Para la documentación y presentaciones: MS Word y MS Power Point
- Para la planificación del proyecto: MS Project
- Para generar los diagramas de clases: ArgoUML
- Para generar los prototipos de pantallas: moqups.com

[5] https://www.linkedin.com/in/philippeombredanne
[6] https://github.com/pombredanne/django-simple-multitenant

- Para el diseño de la base de datos: MySQLWorkbench
- Para el entorno de ejecución: xampp (MySQL, Apache web server, phpmyadmin)
- Para el versionado y control de la configuración: Git (Github)
- Para grabación de la demo: Camtasia Studio

Modelo de Base de datos

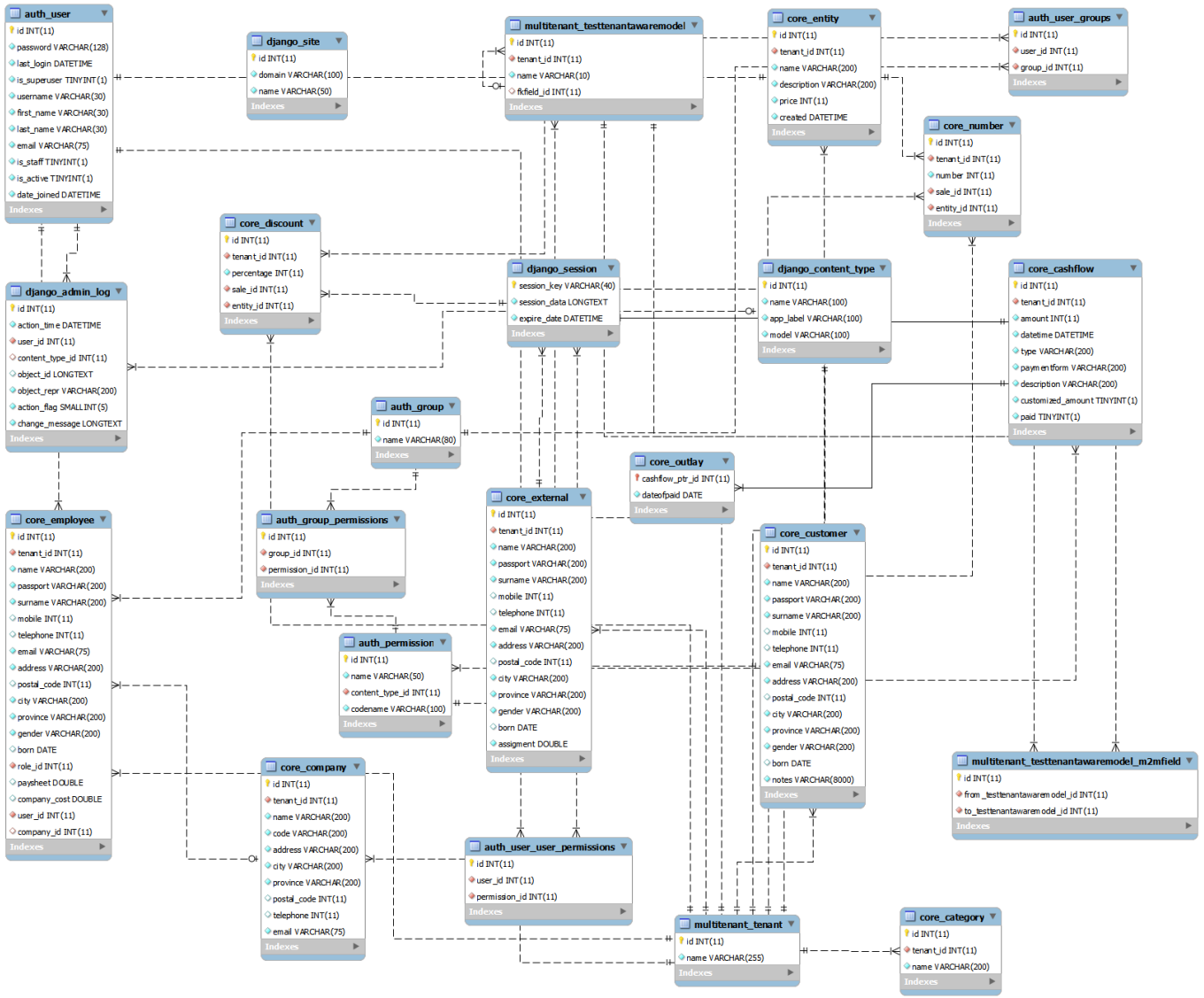


Ilustración 17: Modelo de base de datos

Capítulo 4 : Implementación

Configuración del entorno de desarrollo

Integrated Development Environment (IDE)

Para el desarrollo del software se ha utilizado el IDE eclipse[7], concretamente la versión Juno Service Release 2. Este software se puede descargar de forma gratuita desde http://www.eclipse.org/.

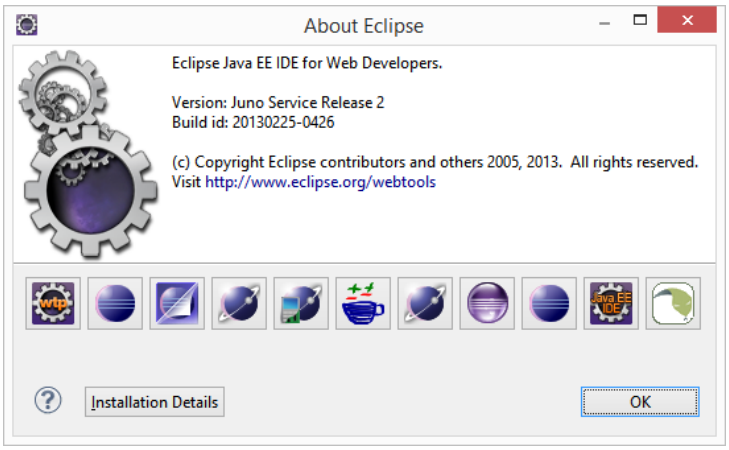


Ilustración 18: Eclipse about

Además se han instalado los siguientes plugins adicionales:

- PyDev - Python IDE for Eclipse, proporcionado por Appcelerator. Se ha decidido la utilización de este plugin dada la facilidad que proporciona para trabajar en entornos Python.
- EGit – Git Team Provider, proporcionado por Eclipse.org. Permite trabajar con proyectos gestionados por git.

[7] http://www.eclipse.org

Ilustración 19: Eclipse plugins

Base de datos y servidor Web

La base de datos utilizada ha sido MySQL y la versión utilizada 5.5.34 - MySQL Community Server (GPL).

El servidor web instalado ha sido Apache/2.4.7 (Win32) OpenSSL/0.9.8y PHP/5.4.22

Para facilitar la gestión de la base de datos y el servidor web se ha optado por la instalación de XAMPP versión 3.2.1 [Mayo de 2013] una distribución de Apache completamente gratuita y fácil de instalar que contiene los servicios indicados y funcionalidades para facilitar su gestión.

Para la gestión de la base de datos se ha utiliza phpmyadmin

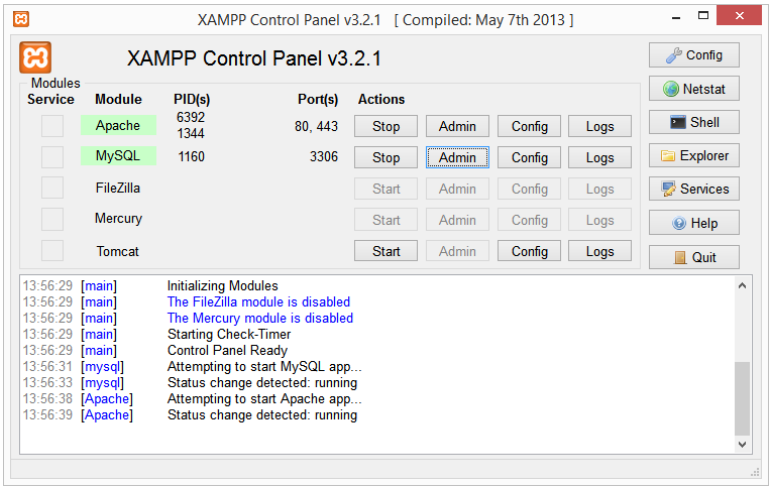


Ilustración 20: XAMPP Control Panel

Intérprete de Python

Se ha utilizado la versión 2.7.6 del intérprete de Python[8].

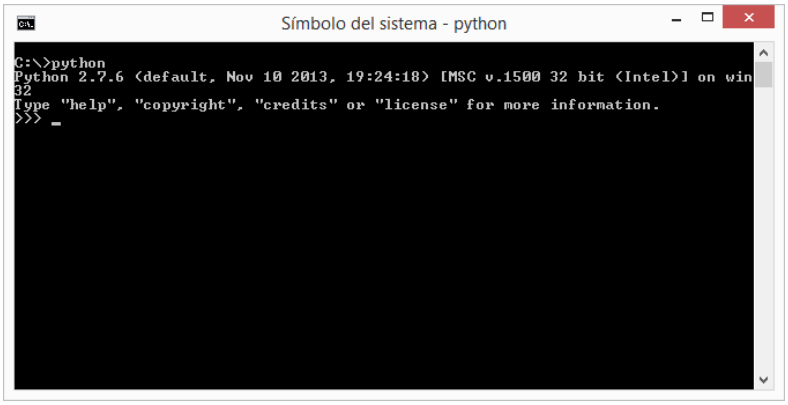


Ilustración 21: python console

Librerías adicionales

Para permitir la comunicación con la base de datos, ha sido necesaria la instalación del módulo de Python: MySQLdb versión 1.2.4[9]

Para las traducciones ha sido necesario instalar las utilidades gettext-utils[10].

Gestión del código fuente

Para la gestión del código fuente se ha utilizado git[11]. Git es un sistema de control de versiones gratuito y distribuido.

Su carácter distribuido permite trabajar en local contra un repositorio de código local.

A su vez, el código fuente se ha subido a Github[12], una forja para alojar proyectos utilizando el sistema de control de versiones Git.

Para trabajar con mayor facilidad con git y github se ha utilizado la herramienta github for Windows[13].

Detalles de la implementación

Estructura del código fuente

El nombre de la aplicación desarrollada para validar el diseño es sme, la cual son las siglas de Small Medium Enterprise.

El primer nivel de la estructura del código está formado por:

[8] https://www.python.org/
[9] https://pypi.python.org/pypi/MySQL-python/1.2.4
[10] http://www.gnu.org/software/gettext/
[11] http://git-scm.com/
[12] https://github.com/jlbarrera/sme
[13] https://windows.github.com/

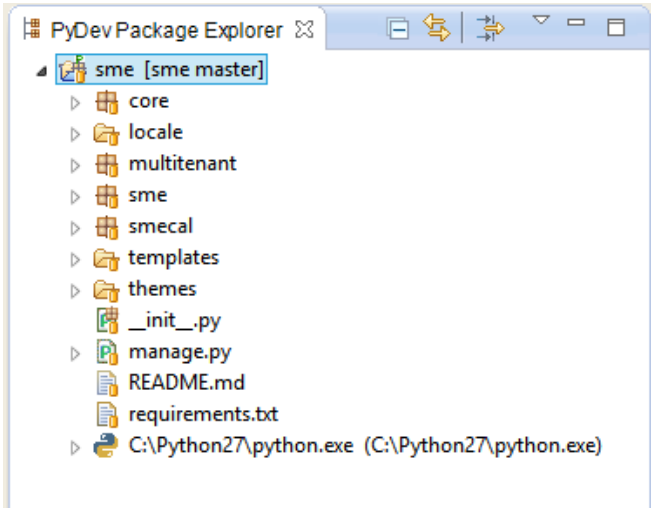


Ilustración 22: Código fuente eclipse

- Core. Es la aplicación donde se encuentra el código del núcleo del sistema.
- Locale. Es el lugar donde se encuentran los ficheros de traducción de la aplicación para la internacionalización de la misma
- Multitenant. Es la aplicación que gestiona el *multitenancy*
- Sme. Es el proyecto en sí, que engloba a todas las aplicaciones y donde se encuentran los ficheros de configuración, administración y construcción de urls entre otros.
- Smecal. Es la aplicación que implementa el módulo de citas.
- Templates. Es el lugar donde se almacenan los templates de la aplicación.
- Themes. Es el lugar donde se almacenan los temas gráficos de la aplicación, incluidos los css, javascript e imágenes.
- Manage.py se trata del script para la gestión de django.
- En requirements.txt se encuentran los requisitos del sistema en cuanto a librerías externas, estos son: Django==1.6 y MySQL-python==1.2.3

Como podemos observar, siguiendo los requisitos de **modularidad** que se habían establecido, existe un núcleo de aplicación que se implementa en sme/core y una serie de módulos añadidos que aumentan las funcionalidades de la aplicación y que se implementan en el sistema sin modificar el core de la aplicación. En este caso, se ha implementado parte del módulo de citas en sme/smecal. En el caso de que se quisieran implementar más módulos como el de productos o el de servicios, se haría de la misma forma en sme/products o sme/services respectivamente.

Para crear un módulo sólo habría que ejecutar el siguiente comando:

```
python manage.py startapp “nombre del módulo”
```

Y comenzar su desarrollo, pudiendo reutilizar tanto el core como el resto de módulos disponibles de la aplicación.

Configuración de la aplicación

La configuración de la aplicación se lleva a cabo desde el sme/settings.py, aquí podremos

configurar todas las opciones de la aplicación como:

La configuración de base de datos:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'test',
        'USER': os.environ.get('MYSQL_USER'),
        'PASSWORD': os.environ.get('MYSQL_PASSWORD'),
        'HOST': '127.0.0.1',
    }
}
```

Las aplicaciones instaladas:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.humanize',
    'multitenant',
    'core',
    'smecal',
    'django.contrib.admin',
    'django.contrib.admindocs',
)
```

Los lenguajes configurados han sido los siguientes:

```
LANGUAGES = (
  ('es', _('Spanish')),
  ('en', _('English')),
)
```

O la ruta a los diferentes recursos como los temas:

```
MEDIA_ROOT = BASE_DIR+"/../themes/"
```

Implementación del modelo

La implementación del modelo del core de la aplicación se lleva desde sme/core/models.py, desde aquí se implementa el modelo definido en el análisis.

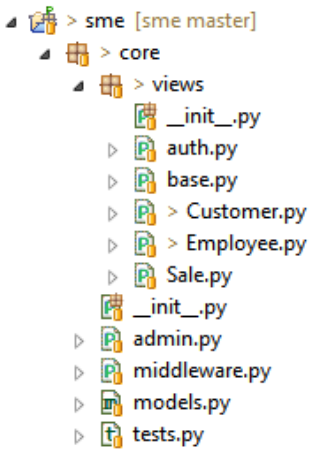


Ilustración 23: Código fuente core

Por ejemplo, la implementación de la entidad Customer sería la siguiente:

```
class Customer(Person):
    notes = models.CharField(max_length=8000, null=True, blank=True)

    def __unicode__(self):
        return '%s %s' % (self.name, self.surname)
```

Como podemos observar tal y como estaba definido en el modelo hereda de Person que sería:

```
class Person(TenantModel):
    name = models.CharField(max_length=200, unique = True)
    passport = models.CharField(max_length=200, null=True, blank=True)
    surname = models.CharField(max_length=200, null=True, blank=True)
    mobile = models.IntegerField(blank=True, null=True)
    telephone = models.IntegerField(blank=True, null=True)
    email = models.EmailField(blank=True)
    address = models.CharField(max_length=200, null=True, blank=True)
    postal_code = models.IntegerField(blank=True, null=True)
    city = models.CharField(max_length=200, null=True, blank=True)
    province = models.CharField(max_length=200, null=True, blank=True)
    gender = models.CharField(max_length=200, null=True, blank=True, choices=GENDER)
    born = models.DateField(blank=True, null=True)

    class Meta:
        abstract = True
```

A su vez, la implementación del modelo del módulo de citas se encuentra en sme/smecal/models.py, cuya implementación para el caso de las citas sería:

```
class Entity(TenantModel):
    name = models.CharField(max_length=200, null=True, blank=True)
    description = models.CharField(max_length=200, null=True, blank=True)
    price = models.IntegerField()
    created = models.DateTimeField(auto_now=True)
```

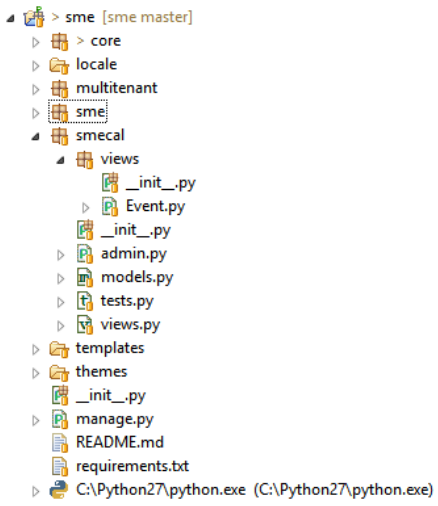


Ilustración 24: Código fuente citas

Implementación de las vistas

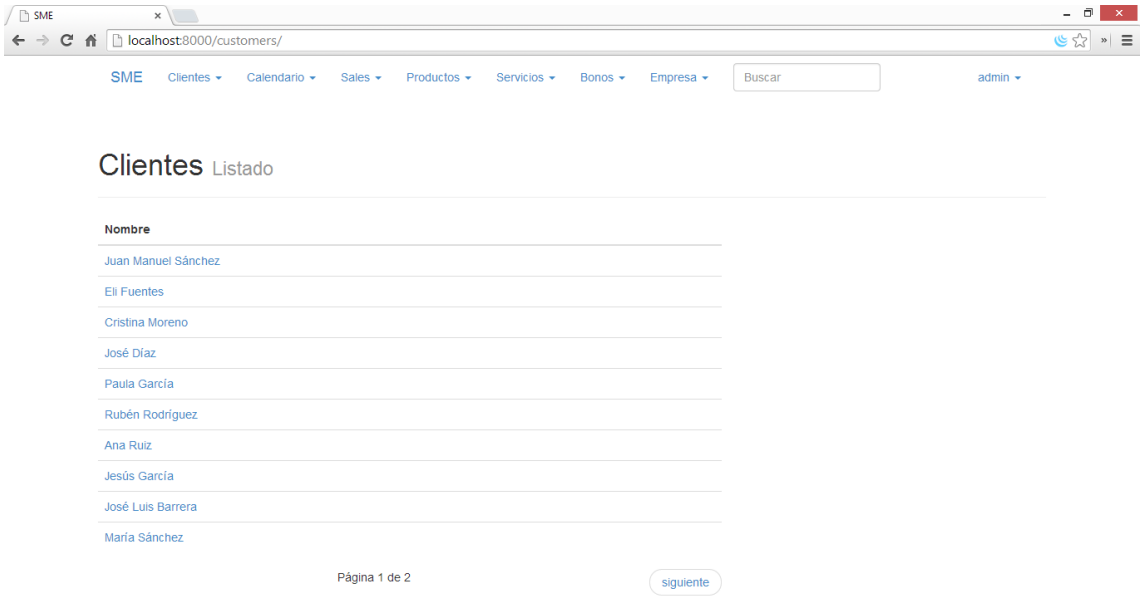
Una vista es una visualización en la aplicación. Las vistas se encuentran dentro de sme/core/views para el caso del núcleo de la aplicación y en sme/smecal/views para el caso del módulo de citas.

Para el caso concreto de la vista de listado de clientes, ésta se implementa en sme/core/views/Customer.py de la siguiente forma:

```
class CustomerList(ListView):
    paginate_by = 10

    def get_queryset(self):
        return Customer.tenant_objects.all().order_by('id').reverse()
```

Este código genera la siguiente pantalla:



Templates

Las vistas utilizan templates para presentar la información en el navegador. Los templates son ficheros HTML con etiquetas que se almacenan en la carpeta templates. Siguiendo con el ejemplo anterior, el template de la vista del listado de clientes se encontraría en templates/core/customer_list.html

```
{% extends "core/person_list.html" %}
{% load i18n %}
{% block section %}
<div class="page-header">
    <h1>{% trans "Customers" %} <small>{% trans "List" %}</small></h1>
</div>
{% endblock %}
```

Gran parte es heredado de core/person_list.html, el cual contiene la tabla que muestra los registros en base de datos de los clientes:

```
{% extends "core/base.html" %} {% load i18n %} {% block content %}
<table class="table table-hover">
        <thead>
                <tr>
                        <th>{% trans "Name" %}</th>
                </tr>
        </thead>
        <tbody>
                {% for person in object_list %}
                <tr>
                        <td><a href="{{person.id}}">{{ person }}</a></td>
                </tr>
                {% endfor %}
        </tbody>
</table>

{% if is_paginated %}
<ul class="pager">
        <li class="previous">{% if page_obj.has_previous %} <a
                href="?page={{ page_obj.previous_page_number }}">{% trans
                        "previous" %}</a> {% endif %}
        </li>
        <span class="page-current"> {% trans "Page" %} {{
                page_obj.number }} {% trans "of" %} {{ page_obj.paginator.num_pages }}
        </span>
        <li class="next">{% if page_obj.has_next %} <a
                href="?page={{ page_obj.next_page_number }}">{% trans "next" %}</a> {%
                endif %}
        </li>
</ul>
{% endif %} {% endblock %}
```

Como se puede observar, se hereda de un template core/base.html que contiene la estructura de la aplicación (menú principal, buscador, etc.) y que es incluida en todas las vistas con el objetivo de reutilizar código y disponer de un sistema de fácil mantenimiento del código.

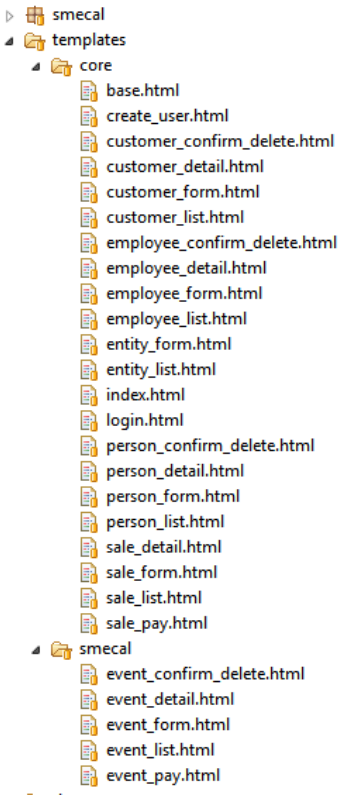


Ilustración 25: código fuente templates

Configuración de URL's

Las direcciones desde las que se accede a las diferentes vistas creadas en la aplicación se configuran desde sme/urls.py mediante expresiones regulares, para el ejemplo del listado de clientes sería:

```
url(r'^customers/$', CustomerList.as_view(), name='customers-list'),
```

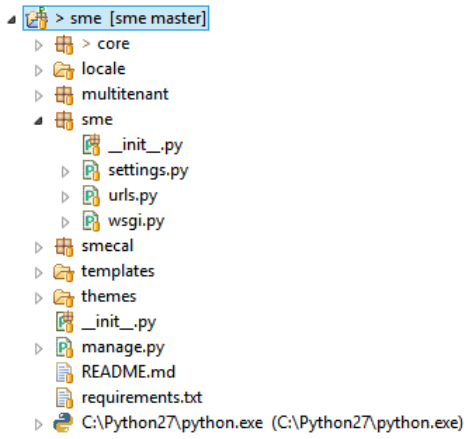


Ilustración 26: código fuente sme

Multiidioma

La aplicación está diseñada desde un inicio para soportar múltiples lenguajes y puede ser traducida a otro lenguaje nuevo sin necesidad de modificar el código fuente de la aplicación. El lenguaje por defecto es el inglés, aunque a modo de validación de esta funcionalidad se ha traducido al castellano.

Para que la aplicación sea multiidioma, el texto debe estar identificado en el código de la siguiente forma:

- En el caso de los templates, mediante la etiqueta *trans*:

```
<h1>{% trans "Customers" %} <small>{% trans "List" %}</small></h1>
```

- En el resto del código mediante _ que se trata de la función ugettext_lazy

```
GENDER = (
          ('M', _("Male")),
          ('F', _("Female"))
    )
```

Esto hace que de forma automática Django recorra el código en busca de estas marcas y nos genere un fichero ".po" en el cual únicamente tendremos que realizar las traducciones. El fichero para el idioma español se encuentra en sme/locale/es/LC_MESSAGES/django.po.

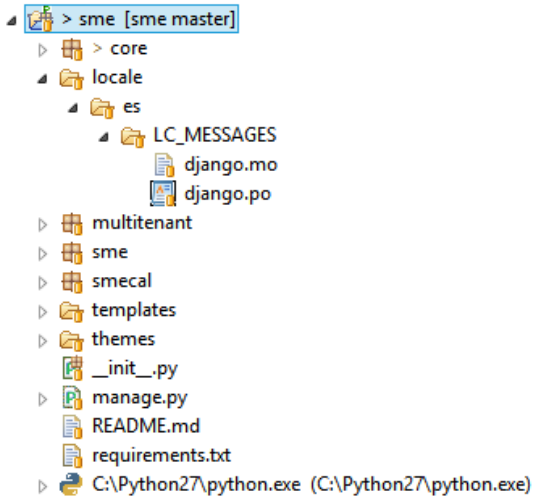


Ilustración 27: código fuente multiidioma

Un ejemplo de traducción sería el siguiente:

```
#: .\core\models.py:73
msgid "Enter Description"
msgstr "Introduce descripción"
```

La identificación de la cadena “Enter Description” en el fichero core/models.py en la línea 73 se hace de forma automática mediante el comando:

python manage.py makemessages -l es

Multitenancy

Todas las entidades modeladas en el sistema están gestionadas por una meta-entidad que controla la propiedad de los contenidos. El código fuentes es implementado en sme/multitenant:

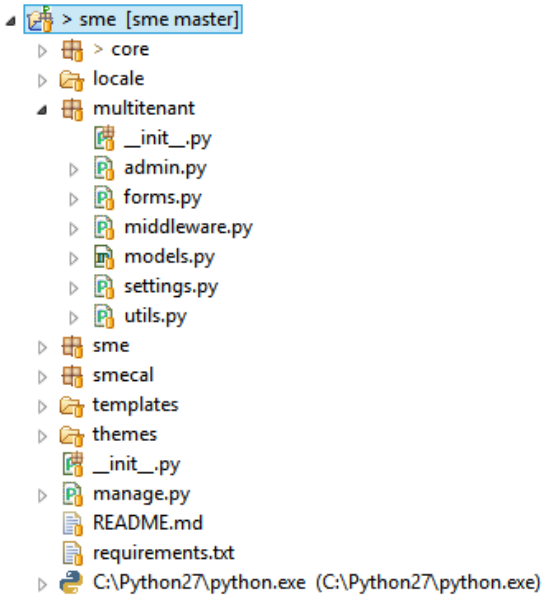


Ilustración 28: código fuente multitenancy

Cada usuario del sistema, es decir, cada empleado, dispone de un **número de identificación de Tenant**. Por ejemplo, para el usuario "admin" el número de Tenant será el 2 (tenant_id) tal y como podemos ver en la tabla de empleados:

	id	tenant_id	name
Editar Copiar Borrar	1	2	admin
Editar Copiar Borrar	9	8	employee1

Ilustración 29: Detalle tabla empleados

Cuando se crea un empleado en la aplicación, automáticamente se crea un Tenant asociado a dicho usuario, si se crea un empleado asociado a una empresa que ya existe, se asocia el Tenant existente a dicho empleado.

A cada entidad se le añade un atributo tenant_id que se completa automáticamente en función del usuario que crea un contenido. A continuación mostramos un ejemplo con datos ficticios de la tabla Customers, en la cual se puede observar que hay 14 clientes, de los cuales 11 pertenecen al Tenant 2 (admin) y 3 al Tenant 8 (employee1):

	id	tenant_id	name	passport	surname
Editar Copiar Borrar	1	2	Sara		Romero
Editar Copiar Borrar	2	2	María	NULL	Sánchez
Editar Copiar Borrar	3	8	Luis	NULL	Romero
Editar Copiar Borrar	4	2	José Luis	NULL	Barrera
Editar Copiar Borrar	5	2	Jesús	NULL	García
Editar Copiar Borrar	6	2	Ana	NULL	Ruiz
Editar Copiar Borrar	7	2	Rubén	NULL	Rodríguez
Editar Copiar Borrar	8	2	Paula	NULL	García
Editar Copiar Borrar	9	2	José	NULL	Díaz
Editar Copiar Borrar	10	2	Cristina	NULL	Moreno
Editar Copiar Borrar	11	2	Eli	NULL	Fuentes
Editar Copiar Borrar	12	2	Juan Manuel	NULL	Sánchez
Editar Copiar Borrar	13	8	Pablo	NULL	Vergara
Editar Copiar Borrar	14	8	Marta	NULL	Salado

Ilustración 30: Detalle tabla clientes

Este identificador es utilizado posteriormente para mostrar a cada usuario únicamente aquellos contenidos que le pertenecen. Si accedemos a la aplicación como admin y mostramos el listado de clientes nos aparece:

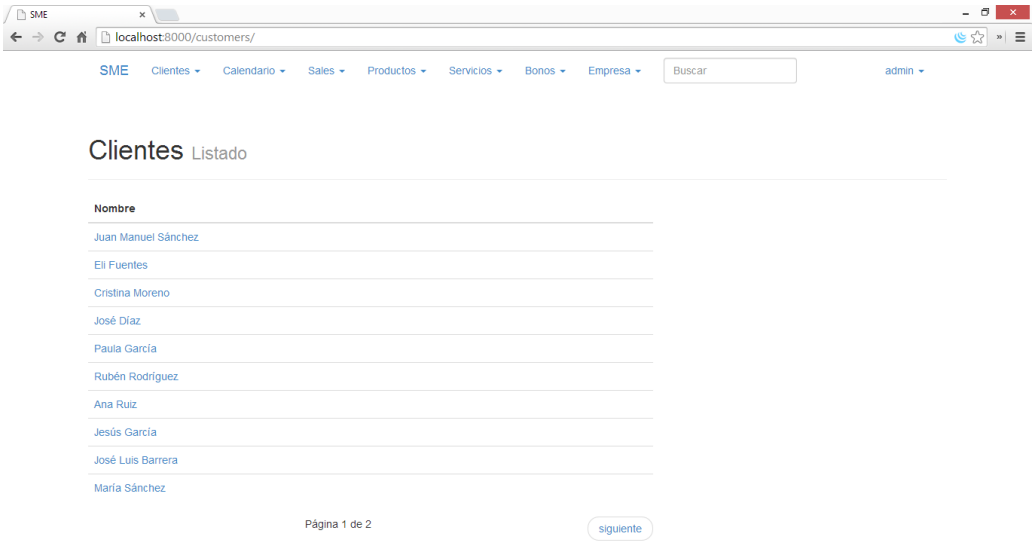


Ilustración 31: Captura listado clientes admin

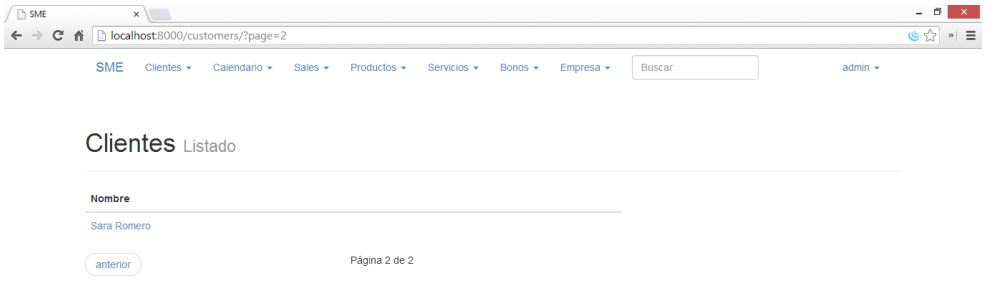


Ilustración 32: Captura listado clientes admin

Por el contrario accedemos ahora con el usuario employee1 al listado de clientes:

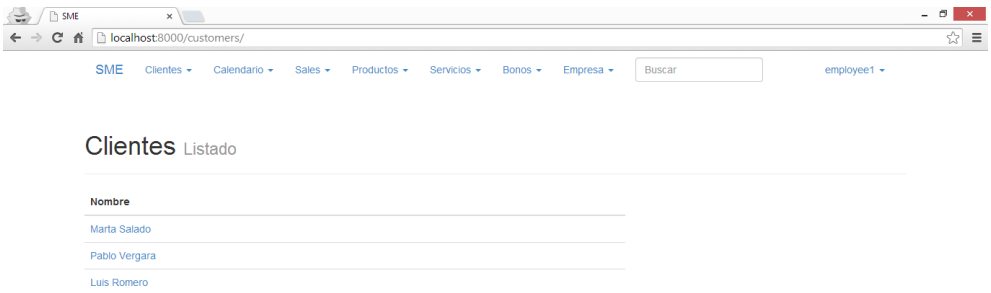


Ilustración 33: Captura listado clientes employee1

Podemos observar que a cada usuario se le muestra sólo los contenidos que le corresponden, a pesar de que todos los contenidos se encuentren almacenados en la misma base de datos y tablas. Esto sucede con todas las entidades creadas en el sistema (citas, productos, servicios, bonos, empleados,…) de forma transparente para el usuario de la aplicación y no añade mayor complejidad al desarrollo, ya que el módulo multitenant abstrae al desarrollador de toda la complejidad existente.

Middleware

Todas las peticiones a la aplicación son interceptadas por una serie de procesos denominados Middleware. En la aplicación se han creado dos:

- LoginRequiredMiddleware se encuentra en sme/core/middleware.py y se ejecuta antes de procesar cualquier lógica de negocio de la aplicación para controlar que todos

los accesos son de forma autenticada. Excepto en una serie de URLs que se almacenan en LOGIN_EXEMPT_URLS en el fichero de configuración, para las cuales no se requiere autenticación.

- Multitenant Middleware se encuentra en sme/multitenancy/middleware.py recupera el Tenant de la *request* para posteriormente hacer el filtrado o asignar el tenant_id.

Interfaz de usuario

La interfaz de usuario se implementa en los templates de la aplicación, para ello se han utilizado los prototipos del diseño del análisis, así como el framework Bootstrap. El cual dispone de diversos componentes y estilos disponibles que facilitan la creación de interfaces web.

Además se ha utilizado la librería jQuery la cual proporciona un conjunto de funcionalidades muy útiles para trabajar mediante javascript.

Estas librerías no se incluyen en el código fuente, sino que son descargadas e integradas en el código mediante CDN (Content Delivery Network[14]).

Pantallas de la aplicación

Pantalla: Inicio de sesión

La autenticación tiene lugar en sme/core/views/auth.py, concretamente en el método login_view. Este método controla la vista de inicio de sesión que se muestra a continuación:

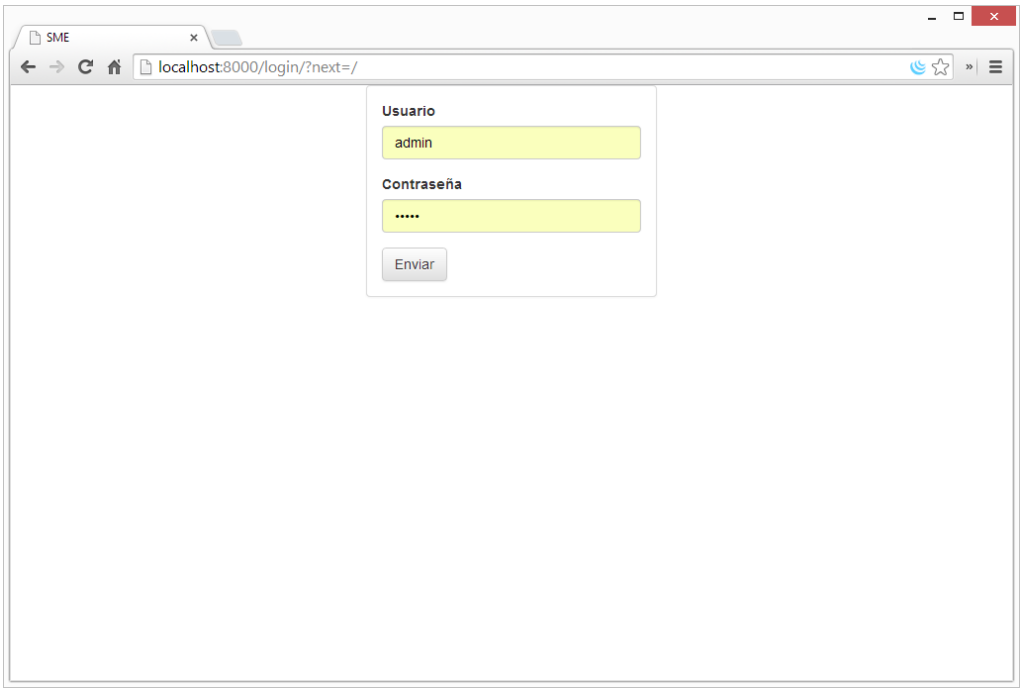


Ilustración 34: Pantalla de Inicio de sesión

[14] http://en.wikipedia.org/wiki/Content_delivery_network

Pantalla: Listado de clientes

Muestra un listado de 10 registros con paginación de la entidad Customer.

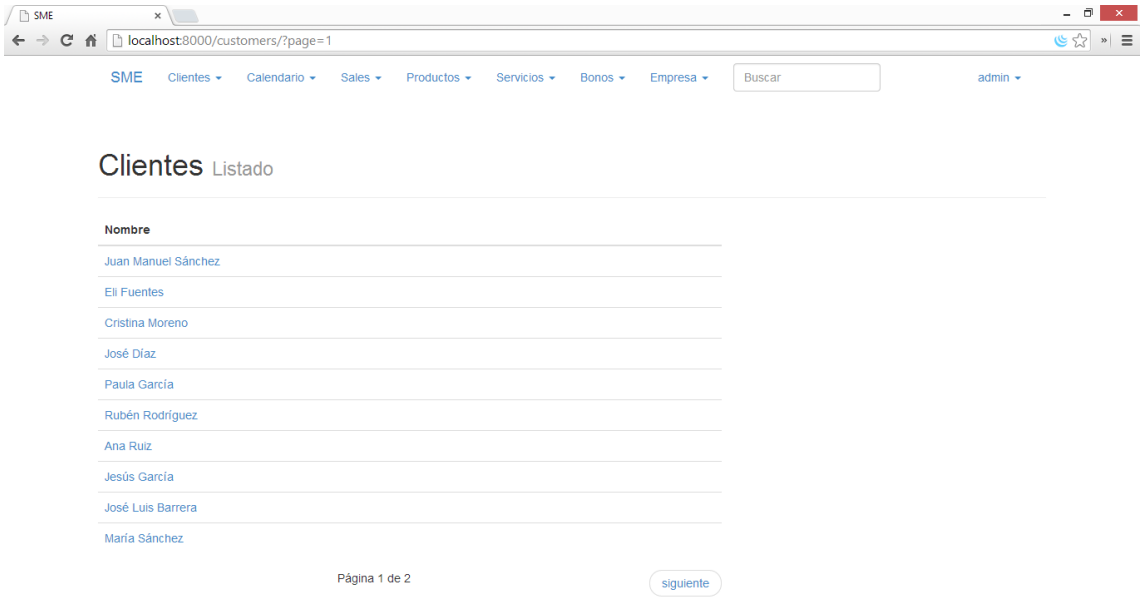


Ilustración 35: Listado de clientes

Pantalla: Detalle de cliente

Actualmente, y a modo validación, sólo se muestra el nombre del cliente en la vista detalle, no obstante el resto de campos están implementados y sólo sería necesario añadirlos a la plantilla para visualizarlos.

Implementación actual:

```
<dl class="dl-horizontal">
  <dt>{% trans "Name" %}</dt>
  <dd>{{ object.name }}</dd>
</dl>
```

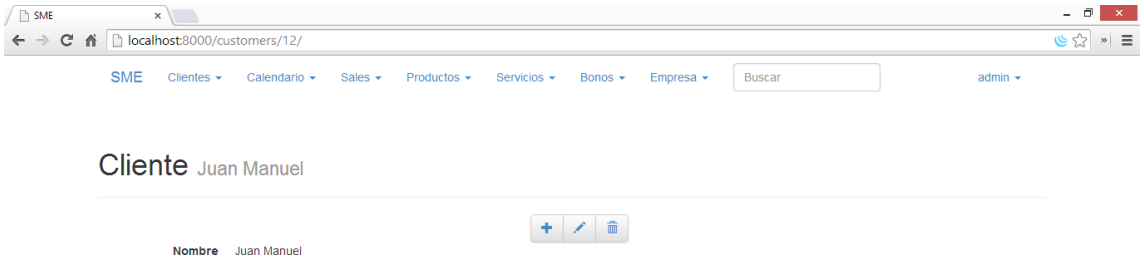


Ilustración 36: Detalle de cliente

Pantalla: Añadir/Editar cliente

Al igual que en la pantalla de detalle, se han seleccionado un conjunto de campos de la entidad Customer para mostrarlos en el formulario de edición. Añadir nuevos campos sería tan sencillo como incluirlos en la variable fields en la view de edición:

```
fields = ['name', 'surname']
```

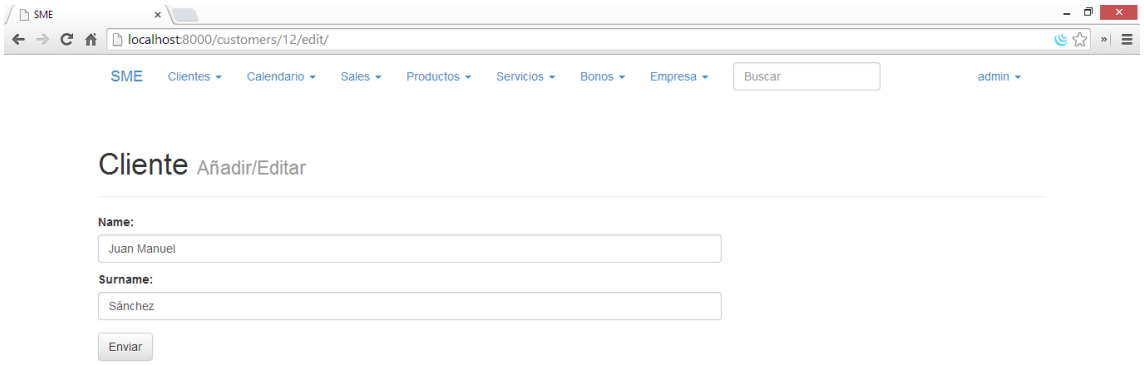


Ilustración 37: Añadir/Editar cliente

Pantalla: Listado de citas

El listado de citas muestra una tabla de 10 registros y paginación, con los campos fecha de la cita, cliente y precio. Además también muestra un campo en forma de icono para el control del pago de la cita. Si aparece el símbolo del €, indica que la cita no se ha abonado aún y se hace clic en dicho icono se accede a la pantalla de pago.

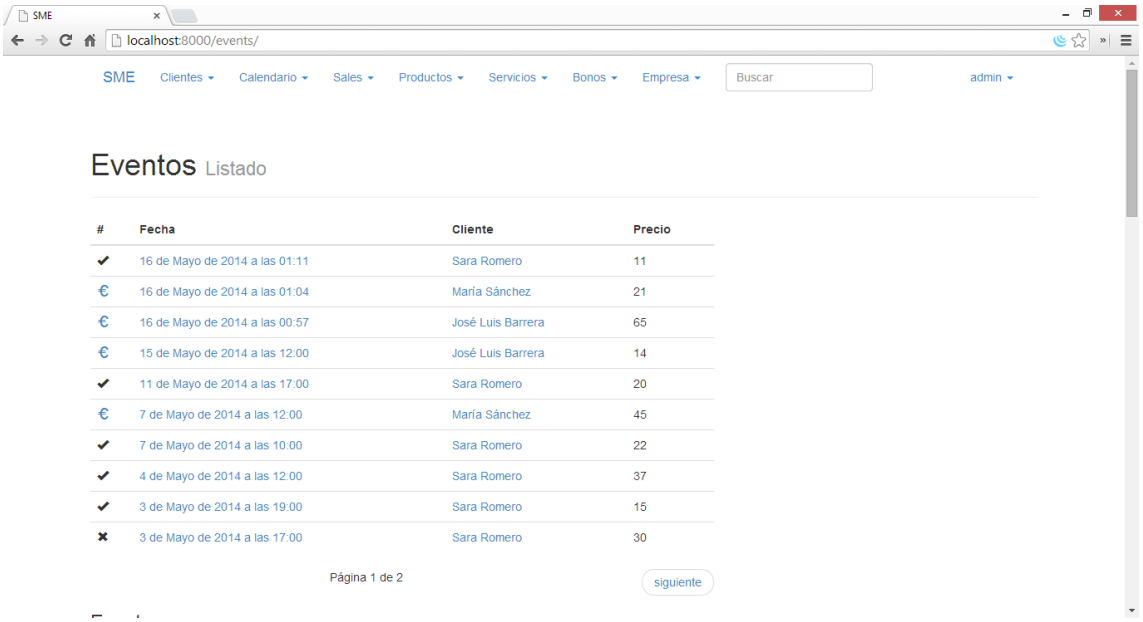


Ilustración 38: Listado de citas

Pantalla: Detalle de cita

En el detalle de la cita se observa el cliente que ha pedido la cita, el precio y la fecha y la hora.

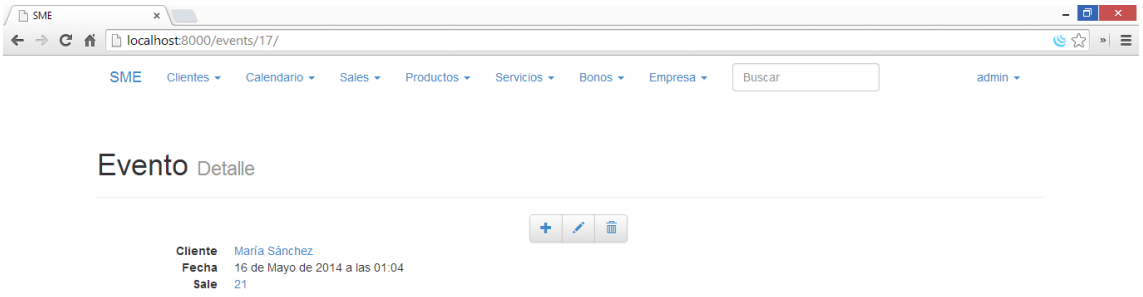


Ilustración 39: Detalle de cita

Pantalla: Añadir/Editar de cita

Se muestra un formulario con los campos necesarios para crear una cita. El cliente y el empleado deben existir previamente, por lo que se muestra un desplegable para su selección.

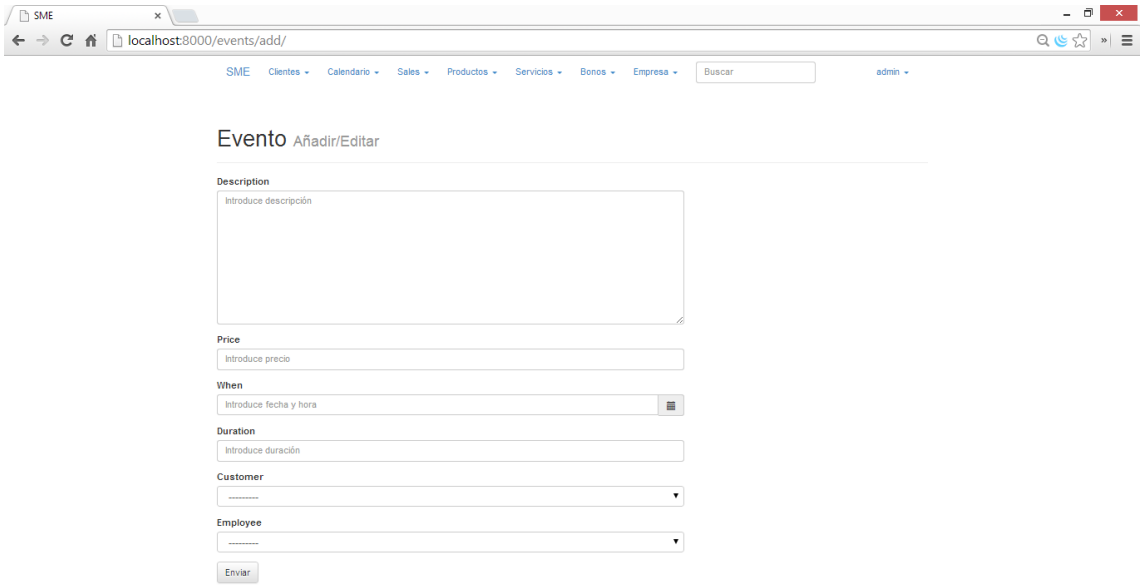


Ilustración 40: Añadir/Editar cita

Pantalla: Listado de ventas

Muestra una tabla listado con 10 registros de ventas. Actualmente las ventas sólo soportan la entidad citas (Event).

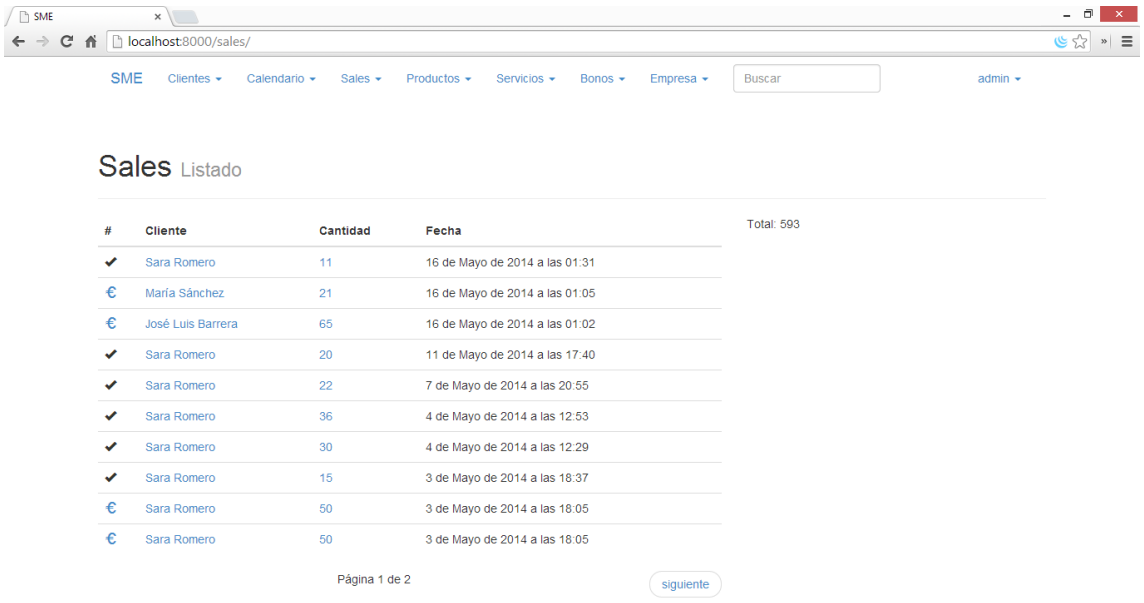


Ilustración 41: Listado de ventas

Pantalla: Detalle de venta

En el detalle de una venta podemos ver el cliente, el precio de venta, así como la entidad a la que hace referencia la venta.

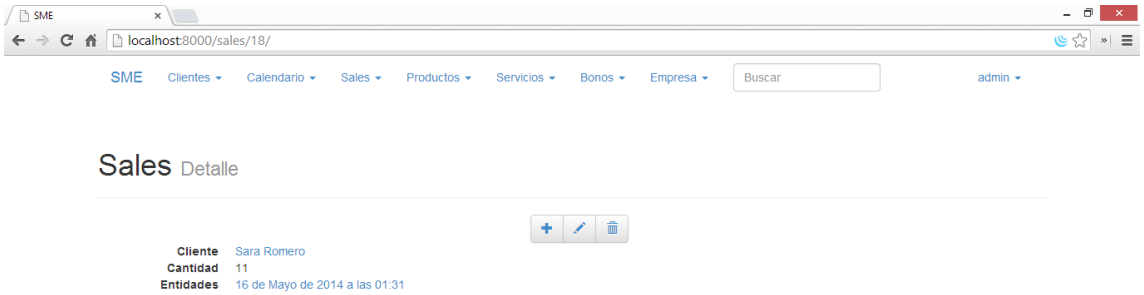


Ilustración 42: Detalle de venta

Pantalla: Listado de empleados

Muestra un listado de empleados para una determinada empresa. Los empleados de una misma empresa dados de alta en la aplicación, comparten el tenant_id.

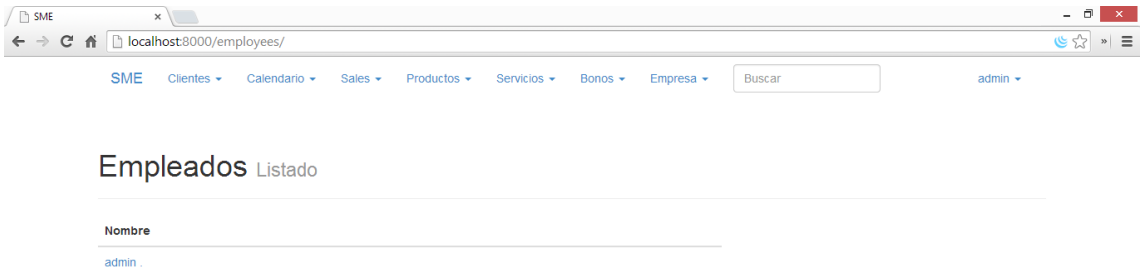


Ilustración 43: Listado de empleados

Pantalla: Administración de django

A modo informativo se muestra el intefaz de administración de django que ha sido habilitado

para facilitar la manipulación de entidades.

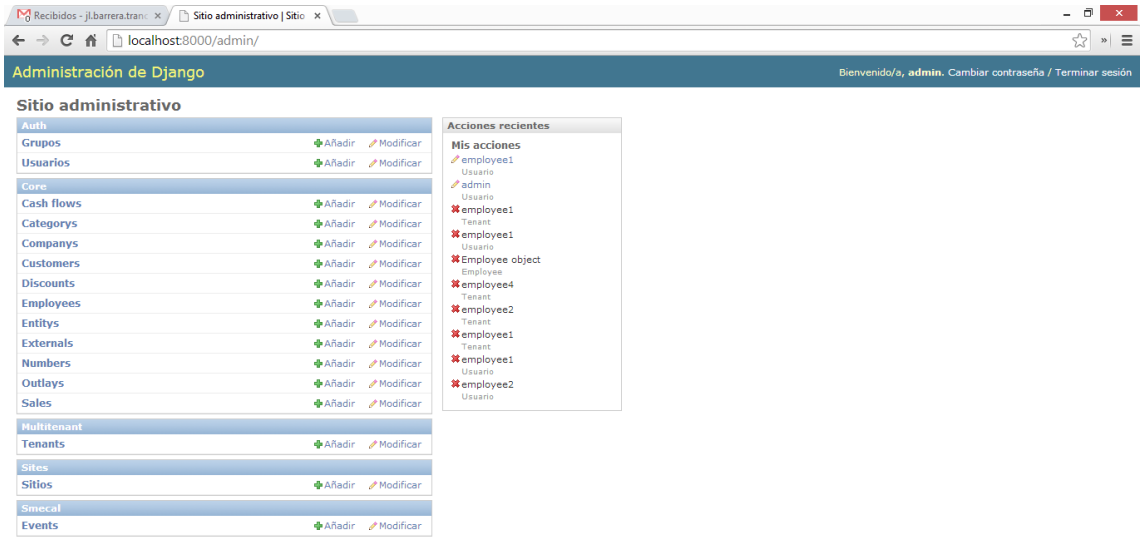


Ilustración 44: Administración de django

Resumen funcionalidades implementadas

A continuación se muestra una tabla resumen con las funcionalidades contempladas en el análisis y aquellas que han sido implementadas a modo de validación tanto del propio análisis como del diseño:

Funcionalidad	Implementación
Arquitectura básica	X
Modularidad	X
Multitenancy	X
Multiidioma	X
Autenticación de usuarios	X
Núcleo de la aplicación	X
Modelado de entidades del núcleo	X
Ventas, Clientes, Empleados	X
Módulo de citas	X

Requisitos cubiertos por la implementación

Código de requisito	Grado de avance
RF-001	100%
RF-002	40%
RF-003	100%
RF-004	0%
RF-005	0%
RF-006	0%
RF-007	0%
RF-008	90%
RF-009	50%

RF-010	80%
RF-011	0%
RF-012	0%
RF-013	0%
RNF-001	100%
RNF-002	100%
RNF-003	100%
RNF-004	100%
RNF-005	100%
RNF-006	100%
TOTAL	**55%**

Capítulo 5 : Valoración económica

El coste de implementación de este proyecto estaría centrado en las horas de desarrollo necesarias para completar las funcionalidades de la aplicación. El coste de adquisición del software es cero, ya que todo el software utilizado es *open source* y gratuito.

A continuación se muestra una estimación de los costes de implementación del sistema diseñado en el presente proyecto:

CONCEPTO	HORAS DESARROLLO
ENTORNO DESARROLLO	8
CONFIGURACIÓN DE LA APLICACIÓN	8
MULTITENANT	24
MULTIIDIOMA	8
INTERFAZ USUARIO	24
NÚCLEO DE LA APLICACIÓN	112
CLIENTES	24
EMPLEADOS	24
EMPRESAS	16
VENTAS	32
GASTOS	16
MÓDULO DE PRODUCTOS	64
PRODUCTOS	24
PROVEEDORES	16
CONTACTOS	16
FAMILIA DE PRODUCTOS	8
MÓDULO DE CITAS & BONOS	64
CITAS	32
BONOS	32
MÓDULO DE SERVICIOS	40
SERVICIOS	24
FAMILIA DE SERVICIOS	16
PRUEBAS	24
DESPLIEGUE EN PRODUCCIÓN	24
TOTAL HORAS	**400**

Capítulo 6 : Conclusiones

El TFC supone la culminación de años de trabajo y esfuerzo, donde se ponen en práctica los conocimientos adquiridos en el área elegida, en este caso el de Ingeniería del Software.

Este proyecto ha supuesto una experiencia personal muy enriquecedora por varios motivos. En primer lugar por el reto personal, tanto por lo ambicioso del proyecto, como por el reto tecnológico de utilizar tecnologías relativamente nuevas para mí. En este sentido el TFC me ha permitido, aprender nuevas tecnologías (como ha sido django), aprender a utilizar nuevos sistemas de gestión del código fuente (como ha sido git), a utilizar las últimas tecnologías en frameworks de UI (como ha sido el caso de bootstrap), así como poner en práctica los conocimiento adquiridos de ingeniería del software durante la carrera.

Por otro lado, me ha permitido profundizar en nuevas arquitecturas de aplicaciones web orientadas a modelos de explotación del software como servicio, a arquitecturas MTV no utilizadas anteriormente y a diseñar desde cero un sistema modular y extensible.

Desde el punto de vista de los resultados del proyecto, destacaría la flexibilidad del framework django, la facilidad de uso y la productividad que proporciona. La documentación existente en la web oficial es completa y de buena calidad. Django está diseñado para favorecer la reutilización del código, hecho que también se ha aprovechado en este proyecto y ha sido determinante para su elección.

Bajo mi punto de vista, la arquitectura de la aplicación implementada sienta las bases sobre las cuales se podría comenzar a desarrollar cualquier tipo de aplicación orientada a la empresa. Obviamente queda mucho trabajo por hacer, y la intención es seguir evolucionando el software y mantenerlo “vivo”, además, el código fuente está publicado en github para que cualquiera que esté interesado pueda hacer sus aportaciones o aprovechar el trabajo realizado. Gracias a esta plataforma para compartir código, el propio proyecto podrá aprovecharse de las aportaciones de terceros si fuera el caso.

Glosario

Cloud Computing. Es un paradigma que permite ofrecer servicios de computación a través de Internet.

SaaS. Software como Servicio (del inglés: Software as a Service, SaaS) es un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC), a los que se accede con un navegador web desde un cliente, a través de Internet. La empresa proveedora TIC se ocupa del servicio de mantenimiento, de la operación diaria y del soporte del software usado por el cliente.

Multitenancy. Corresponde a un principio de arquitectura de software en la cual una sola instancia de la aplicación se ejecuta en el servidor, pero sirviendo a múltiples clientes u organizaciones (tenedor o instancia). Este modelo se diferencia de las arquitecturas con múltiples instancias donde cada organización o cliente tiene su propia instancia instalada de la

aplicación. Con una arquitectura de tenencia múltiple, la aplicación puede particionar virtualmente sus datos y su configuración para que cada cliente tenga una instancia virtual adaptada a sus requerimientos.

Tenant. Cliente de una aplicación con arquitectura multitenancy.

MTV (Model, Template and View). Arquitectura en la que M representa "Modelo", la capa de acceso a datos, V representa "View", la capa de lógica del negocio y T representa "template Plantilla", la capa de presentación.

ORM (mapeador objeto-relacional). Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia.

API (Application Programming Interface). Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Integrated Development Environment (IDE). Es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

CDN (Content Delivery Network). Es una red superpuesta de computadoras que contienen copias de datos, colocados en varios puntos de una red con el fin de maximizar el ancho de banda para el acceso a los datos de clientes por la red.

Bibliografía

[01] Inmersión en Python. http://es.diveintopython.net/toc.html

[02] Django Documentation. https://docs.djangoproject.com/en/1.6/

[03] Bootstrap. http://getbootstrap.com/

[04] jQuery API documentation. http://api.jquery.com/

[05] Stack Overflow. http://stackoverflow.com/