SOL: Sistema de Optimización Logística para Banco de Alimentos de Madrid

SOL: System for Optimized Logistics for BAM

UNIVERSIDAD COMPLUTENSE
MADRID

Autor
Enrique Miguel Torrijos Gabriel

Dirigido por
José Luis Vázquez Poletti

Curso
2019 - 2020

Trabajo de Fin de Grado
Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

Dedicatoria

A mi tutor por escucharme y dedicarme su tiempo y experiencia en el proyecto, a mis amigos por acompañarme siempre, a mis padres por guiarme y ayudarme a ser quien soy, a mi cuñada por su apoyo siempre alegre e incansable y a mi hermano por enseñarme lo que es ser un informático y no dudar nunca de lo que puedo alcanzar.

Resumen

El Banco de Alimentos de Madrid (BAM) es una ONG que busca entregar alimentos a todos los que no puedan comprarlos. La organización tiene mucho movimiento de personal ya que la gran mayoría son voluntarios, esto hace difícil una gestión interna del Banco. Además no hacen mucho dinero, el que consiguen, en la mayoría de los casos por donaciones, lo utilizan para comprar y mantener alimentos para entregarlos a la gente que los necesita, por eso sus equipos informáticos son antiguos y deben tener cuidado con las aplicaciones que instalan en ellos, por cualquier incompatibilidad que pueda surgir.

El Sistema de Optimización Logística para el Banco de Alimentos de Madrid (SOL-BAM) es una solución a este problema. La gente se puede conectar a través de un navegador y gestionar la información interna que no controlaban fácilmente. Este Sistema resuelve los siguientes problemas:

- Conocer la asistencia de los voluntarios al Banco (no tienen un horario).
- Reservar las salas de reuniones que tienen para organizarse internamente o con otras empresas.
- Alguna forma de comunicarse entre los empleados y voluntarios y notificar problemas a los informáticos.
- Organizar las visitas de colegios u otras entidades al Banco.

Todo esto lo controla un servidor que recibe todos estos datos, de esta manera se puede usar desde cualquier ordenador independientemente del sistema operativo que tenga instalado.

Palabras Clave

SOL-BAM, SOL, BAM, Banco, Alimentos, Voluntarios, CMS, Joomla, XAMPP

Abstract

The Banco de Alimentos de Madrid (BAM) is a NGO that seeks to deliver food to anyone who can't afford it. This organisation has a lot of changes in personnel because the vast majority of them are volunteers, making very difficult managing the Bank's internal process. In addition to this, they don't make much money, the one they make, is used to buy more food and keep it for anyone who needs it, that's why their computer equipment is outdated and they must be careful with the applications they install on it, for any incompatibility that may arise.

The System for Optimized Logistics for Banco de Alimentos de Madrid (SOL-BAM) is a solution to this problem. People can connect through a browser and manage all this information that wasn't controlled effectively before. This System solves the following problems:

- Know the attendance of volunteers at the Bank (they don't have a schedule).
- Reserve the meeting rooms they have to organize themselves internally or with other companies.
- A way to comunicate between volunteers and employees and notify problems to the IT department.
- Organise the visits of schools or other institutions to the Bank.

Everything is controlled by a server that manage all the data, this way SOL-BAM can be used from any computer regardless of the system installed.

Key Words

SOL-BAM, SOL, BAM, Bank, Food, Volunteers, CMS, Joomla, XAMPP

Índice general

Capítulo 1

Introducción

El Banco de Alimentos de Madrid (BAM) es una ONG que busca entregar alimentos a todos los que no pueden comprarlos. La organización dispone de 4 sedes, Alcorcón, Alcalá, Mercamadrid y San Fernando, que es la principal. Esta aplicación busca satisfacer las necesidades de la sede del colegio San Fernando, cualquier problema aparte que tenga otra sede no se ha contemplado.

El BAM tiene mucho movimiento de personal ya que la gran mayoría son voluntarios, esto hace difícil una gestión interna del organismo. Además no hacen mucho dinero, el que consiguen, en la mayoría de los casos por donaciones, lo destinan en su mayoría a comprar y mantener alimentos para entregarlos a la gente que los necesita, por eso sus equipos informáticos son antiguos y deben tener cuidado con las aplicaciones que instalan en ellos, por cualquier incompatibilidad que pueda surgir.

Un detalle importante de la aplicación es el usuario al que está orientado, la mayoría de ellos es muy mayor, por eso la accesibilidad de la aplicación debe de ser lo más alta posible.

1.1. Introduction

The Banco de Alimentos de Madrid (BAM) is a NGO that sends food to anyone who can't afford it. The organisation has 4 offices, Alcorcón, Alcalá, Mercamadrid and San Fernando, which is the most important. This application seeks to meet the needs of San Fernando school, any specific problem from another site is not addressed.

The BAM has a lot of changes in personnel because the vast majority of them are volunteers, making very difficult managing the bank's internal process. In addition to this, they don't make much money, the one they make, is used to buy more food and keep it for anyone who needs it, that's why their computers are outdated and they need to be careful of which applications install in them, it can cause a lot of incompatibility issues.

An important detail of the system is the target user, the majority of them is very old, this is why the accessibility of the application must be as high as possible.

1.2. Antecedentes

En abril del 2019 se contactó con el profesor José Luis Vázquez Poletti para presentarle una idea para un TFG, consistía en una red social para empresas que relacionaba las demandas de potenciales clientes con proveedores que pudieran cumplirlas y los ponía en contacto. Esta aplicación tenía un tratamiento de la información extenso que buscaba optimizar las relaciones entre las empresas. El problema de esa aplicación es que los datos no iban a ser reales y no resolvía un problema real, era una idea sin ningún contexto real que demostrara su utilidad, así que se acabó descartando.

Después de la presentación de la propuesta, el profesor y el autor del trabajo se pusieron en contacto varias veces para plantear esa idea en un contexto real. Desde hace varios años, algunas asociaciones de la facultad organizaron eventos de videojuegos con varias ONGs, en uno de estos eventos se recogió comida y dinero para el Banco de Alimentos de Madrid. El profesor Vázquez tenía el correo de uno de sus voluntarios regulares, así que se decidió ponerse en contacto con él para proponer a la Fundación una tecnología que implementara la idea original. La aplicación iba a ser la misma, solo que en este caso relacionaba las donaciones de alimentos de diversos donantes con clientes que recibirían esa comida.

Cuando tuvo lugar dicha reunión con el Banco, explicaron que ya tenían una aplicación similar en la compañía, pero se mostraron abiertos a explicar cómo se estructuraba la organización y su proceso interno para encontrar algún problema que se pudiera resolver, de esta manera se desarrollaría algo que les fuera de utilidad. Se hablaron de varias opciones, como gestionar la nómina de los empleados que tienen, un sistema para gestionar su inventario o alguna forma de controlar los voluntarios, pero había que comprobar si algunos de estos casos ya estaban controlados.

Se diseñaron distintas ideas para resolver estos casos y se presentaron al Banco. Varias opciones planteadas ya las controlaban, como la nómina, pero se detectó un detalle muy importante, muchos de los problemas que tenían se podían agrupar como problemas del día a día, eran pequeñas situaciones que sucedían porque no tenían ningún sistema que les permitiera organizarse en su actividad diaria. Después de hablarlo con el responsable del Banco, se propuso agrupar estas situaciones para estudiarlas y diseñar una solución que las resolviera y aceptaron.

1.3. Restricciones

Para diseñar el sistema, había varios detalles que se debían tener en cuenta.

El primero era el usuario al que estaba orientado. Este sistema lo iban a utilizar los voluntarios que trabajaban en las oficinas, que en la mayoría de los casos son personas jubiladas. Esto implicaba que la accesibilidad del sistema debía ser lo más alta posible. También debía ser lo menos invasivo

posible para no limitar el trabajo que hacían los usuarios, ya que, de lo contrario, podían dejar de ir a ayudar.

Esta aplicación solo se quería utilizar en el colegio San Fernando, por lo tanto, cualquier problema específico de otras sedes no se debía contemplar.

Otro restricción importante eran los equipos del Banco. La organización no tenía mucho dinero y la prioridad seguía siendo la comida, así que los equipos eran muy antiguos, los actualizaban gracias a los equipos donados de otras empresas. Todos los equipos tenían instalados un sistema operativo Windows, desde el XP hasta el 10, y a veces se producían problemas de incompatibilidad entre las aplicaciones y equipos. Un factor importante para escoger las aplicaciones que usaban era que tuvieran distintas versiones para sistemas más antiguos como Microsoft Office.

Cómo había un riesgo de que un equipo dejara de funcionar, también se hacía un tratamiento minucioso de la información y sus copias de seguridad, todo estaba controlado desde el servidor de la empresa mientras la gente trabajaba. Cualquier equipo o aplicación nueva debía poder funcionar correctamente mientras se seguían dichos protocolos.

Otra situación que hay que recalcar es la situación económica y los gastos de la Fundación. El Banco ya tenía varios servicios por los que pagaba mensualidades, como el hosting de varias máquinas virtuales, así que cualquier nueva tecnología debía ser lo más barata posible o que pudiera aprovechar algún servicio previo.

Por último, hay que aclarar las fechas del proyecto. El sistema debía estar implementado en mayo o en septiembre, siguiendo los plazos para presentarlo en una de las convocatorias establecidas por la facultad.

1.4. Objetivos

El objetivo principal del proyecto consiste en desarrollar una aplicación que ayude a los empleados y voluntarios con su trabajo en el día a día. Para completarlo se identificaron 5 objetivos que se debían resolver:

- Conocer la asistencia de los voluntarios al Banco para ponerse en contacto con otra persona si el que se buscaba no estaba ese día. En algunos casos la gente buscaba por el edificio a algún responsable o jefe de departamento concreto sin saber que ese día no iba a aparecer, perdiendo mucho tiempo.
- Reservar las salas de reuniones. Como no había nada así, no podían saber con seguridad a que sala tenían que llevar una empresa dispuesta a donar o si tendrían alguna sala disponible. Este sistema también permitiría que nadie entrase en una sala en mitad de una reunión.
- Comunicación entre los empleados y voluntarios. Con un sistema que tenga implementada esta funcionalidad, no necesitan una herramienta externa como WhatsApp, llamarse por teléfono o levantarse de su puesto para hablar entre ellos, agilizando el trabajo.

- Sistema de notificación de incidencias para los técnicos. Cuando un voluntario tenía un problema con algún equipo, llamaba a los técnicos para que lo solucionaran, pero no eran muchos y podían estar ocupados ayudando a otras personas. De esta manera tendrían una herramienta que guardara las incidencias y que se pudieran ver en cualquier momento.
- Organizar las visitas de colegios u otras entidades. Se han dado casos de varias visitas con mucha gente en los almacenes a la vez, impidiendo el flujo normal de la empresa.

1.5. Plan de trabajo

La metodología establecida para realizar este proyecto es cascada o waterfall. Debido a la naturaleza del trabajo, se necesitaba capturar todos los requisitos del Banco antes de analizar qué herramientas se podían utilizar en el desarrollo y esta metodología ofrecía una estructura rígida que se fundamentaba en que no se avanzaba a la siguiente fase hasta que se hubiera terminado la anterior. Las fases del proyecto fueron las siguientes:

- **Obtener y analizar los requisitos del sistema**: Durante los primeros meses del proyecto se tenían reuniones con el banco para recoger los problemas que habían tenido mientras organizaban la comida y preparaban envíos a los comedores sociales. Mientras se capturaba las incidencias se agrupaban según su causa. La mayoría de estas incidencias eran bastante pequeñas y no tenían un efecto destacable en la empresa, así que se descartaban para priorizar los casos más graves. Algunos de estos casos eran bloqueos en algunos trabajos porque el encargado no estaba, como gestionar unos pedidos de otras instituciones, o algún cambio de puesto de algún empleado mientras se revisaba su equipo por algún fallo.

  Al principio de la siguiente reunión se mostraban posibles soluciones para resolver los problemas que se habían mencionado previamente y se debatía sobre cómo mejorar dichas soluciones.

  Estas reuniones siguieron hasta que las nuevas incidencias que surgieron eran idénticas a las que se habían contemplado.

- **Diseñar la arquitectura**: Una vez que se establecieron todos los requisitos y restricciones, se empezó a diseñar la arquitectura de la aplicación. Para escoger el mejor diseño se utilizó un brainstorming para tener muchas opciones. Se estudiaban individualmente y luego se comparaban con el resto, si era peor se descartaba y si era mucho mejor que alguna anterior se sustituía. El diseño de las arquitecturas era de alto nivel, por lo tanto se buscaron algunas herramientas de ejemplo para ver el sistema en gran escala, pero en este punto no se estudiaron las herramientas como tal.

  Cuando se escogió la arquitectura, se empezaron a buscar herramientas que permitieran hacer lo que necesitaba. Algunas de ellas se escogieron debido a tener experiencia previa con ellas, pero en las que no se investigó como funcionaban y qué aportaban al trabajo respecto a otras similares.

- **Desarrollo de SOL-BAM**: Con las herramientas escogidas, se revisaron otra vez los objetivos y se organizaron siguiendo un orden de prioridad establecido por el Banco.

Se empezó con la gestión de los usuarios en el sistema y luego se desarrollaron los objetivos del proyecto siguiendo el orden establecido. Mientras se implementaba un objetivo se hacían pruebas unitarias del nuevo módulo y, cuando se terminaba esa parte, se hacían pruebas de integración comprobando todos los módulos implementados hasta el momento, de esta manera si había un error que afectara a otras partes de la aplicación se detectaría rápidamente y se podría solucionar antes de seguir con el siguiente objetivo.

Durante esta fase se mantuvieron varias reuniones con el Banco para mostrar los avances del proyecto y que confirmaran que lo que se estaba desarrollando era de su agrado. En el caso de que no fuera así, se podía notificar en el momento para cambiarlo.

- **Pruebas finales**: Una vez que se terminó todo el desarrollo se hicieron pruebas del sistema. En estas pruebas se revisó la funcionalidad de los módulos y se comprobó que se podía conectar desde otros dispositivos.

En este proyecto no se contempló la integración del software en los equipos del Banco porque no se conocía en detalle la arquitectura interna de sus sistemas ni sus protocolos de seguridad. Se configuró el proyecto para que funcionara en un entorno sencillo que se pudiera exportar con facilidad a otra arquitectura. Una vez que terminase el proyecto, se podría demostrar el funcionamiento del software en el caso de que se pidiera.

Capítulo 2

Captura de Requisitos

Para escoger una arquitectura eficiente para el proyecto, había que recoger los requisitos del Banco, tanto los funcionales como los técnicos.

Con los requisitos funcionales se recogerían las necesidades que se debían cumplir, identificando los objetivos del proyecto, y con los técnicos podríamos indagar en algunas restricciones que se debían tener en cuenta para que el sistema lo pudiera utilizar el Banco en sus oficinas.

En este capítulo se va a explicar que proceso se siguió para extraer los requisitos y las reuniones que se tuvieron para ello, después se explicaran todos ellos, organizados por el tipo al que pertenecen.

2.1. Extracción de los requisitos

Durante las primeras reuniones con los responsables del Banco se explicaron algunas tecnologías que utilizaban y comentaron por encima su arquitectura para conocer la organización del Banco.

Explicaron el proceso interno de la empresa y el movimiento de comida dentro de la misma, cómo pasaba de los donantes a los comedores sociales y demás instituciones. La comida se mantenía en sus estanterías y se vigilaba su estado periódicamente para no entregar comida en malas condiciones.

Después se revisó el trabajo en las oficinas, enseñaron cómo se ponían en contacto con donantes y clientes y organizaban el inventario. También explicaron cómo hacían la nómina de los empleados y cómo cambiaban los equipos defectuosos.

Una vez que terminaron de explicar todos estos procesos, se observaron las pequeñas interacciones del día a día. Ellos apuntaban pequeñas situaciones que ocurrían y cualquier funcionalidad que querían que se incluyera y las compartían en posteriores reuniones. Después de comunicarlas, se revisaban individualmente para encontrar qué las provocaba, una vez que se detectaba una posible

causa, se comparaba con el resto para asegurarla. Después se enseñaron al Banco para confirmar los problemas que provocaban esas incidencias y ellos afirmaron que estaban de acuerdo con la lista mostrada. Con estas causas, las funcionalidades y la explicación de los procesos se crearon los requisitos de la aplicación, y estos permitieron redactar los objetivos a alcanzar y las restricciones a cumplir.

Se mostraron al Banco los objetivos y las restricciones que se extrajeron para que las validasen y así poder empezar con el diseño de la arquitectura.

2.2. Requisitos funcionales

Estos requisitos recogían lo que se esperaba que resolviera el sistema, más tarde se convirtieron en los objetivos del proyecto:

- Conocer la asistencia de los voluntarios al Banco para ponerse en contacto con otra persona si el que se buscaba no estaba ese día. Esta funcionalidad la pidió el Banco desde el principio.

- Reservar las salas de reuniones. Este requisito surgió porque en algunas reuniones que se mantenían con el Banco algún voluntario entraba en la sala pensando que estaba vacía. Estas incidencias también surgieron en reuniones con otras entidades.

- Comunicación entre los empleados y voluntarios. Este requisito se pidió porque muchas veces la gente tenía en silencio los teléfonos y no se enteraban cuando alguien los llamaba, por lo que los voluntarios empezaron a levantarse de su puesto para informar a otro compañero.

- Sistema de notificación de incidencias para los técnicos. Este problema se detectó al estudiar el proceso de reportar una incidencia. El proceso original consistía en visitar el departamento de IT e informar del fallo. Si había personal te atendían, pero había ocasiones en las que no había nadie.

- Organizar las visitas de colegios u otras entidades. Esta funcionalidad la pidió el Banco en julio del 2020, surgió porque revisaron el histórico de visitas de la empresa y un día se presentaron más de 100 personas a conocer el Banco, una cantidad que no se esperaba y que superaba con creces el límite establecido para organizar el trabajo ese día sin complicaciones. Debido a la fecha en la que se pidió y que era el requisito menos valioso para la Fundación, fue considerado el menos importante.

Con todos estos requisitos implementados, se tendría una solución que cumpliría con todos los objetivos del proyecto, pero había que tener en cuenta los requisitos técnicos para que la aplicación se pudiera instalar en el Banco y funcionase sin provocar nuevas complicaciones.

2.3. Requisitos técnicos

Durante las reuniones se mencionaron una serie de restricciones que se debían tener en cuenta para diseñar la arquitectura del sistema:

- La aplicación debía permitir que se conectaran varios usuarios a la vez desde distintos equipos, de esta manera se podría acceder a la información general y se actualizaría cualquier dato importante que debieran saber el resto sin que afectase al funcionamiento normal del software.
- Los equipos desde los que se iba a acceder al sistema eran en su mayoría antiguos y había un riesgo de que dejasen de funcionar correctamente. La aplicación debía delegar lo menos posible la gestión de la información en los equipos individuales para evitar pérdidas de datos en caso de cualquier problema.
- El sistema debía permitir incluir nuevas funcionalidades sin necesidad de acceder al código fuente de la aplicación. Este era un requisito muy importante para el Banco porque querían un sistema que se pudiera actualizar en el futuro y que pudiera resolver cualquier problema que surgiese más adelante.
- El usuario objetivo era muy mayor, la accesibilidad del sistema tenía que ser lo más alta posible.
- La solución debía ser lo más barata posible para usarla. Este requisito era importante porque el Banco no tenía mucho dinero, dependía por completo de las donaciones.

Con estas restricciones se podía diseñar una aplicación que se adaptara a la organización del Banco, por lo que se resolverían los problemas que tenían sin provocar otros nuevos.

Capítulo 3

Arquitectura del sistema

Antes de programar como tal, había que pensar cómo iba a ser la arquitectura de la aplicación. Una vez que se tuviera un esquema del sistema, se podría investigar qué herramientas iban a ser necesarias para implementarlo.

Teniendo en cuenta los requisitos del proyecto, se escogió implementar un sistema cliente-servidor. Un sistema así permite centralizar todo el proceso del negocio en un solo equipo, los usuarios se conectan a él y el servidor gestiona toda la información. Esta arquitectura permitiría abstraerse de los equipos individuales y del sistema que tuvieran integrado, ya que los usuarios se conectarían a través de un navegador web.

En este capítulo se va a explicar las herramientas que forman parte de la arquitectura del proyecto y el proceso que se siguió para escoger cada una.

3.1. Esquema de las herramientas seleccionadas

Cuando se escogió la arquitectura que iba a utilizar el sistema, se buscaron distintas herramientas qué permitieran diseñar una página web. Se decidió utilizar un Content Management System (CMS) ya que, con este tipo de software, se podía implementar una aplicación que cumpliera todos los requisitos funcionales y técnicos, en especial el del sistema que pueda actualizarse fácilmente con el paso del tiempo. El CMS que se escogió fue Joomla.

Dado que Joomla escuchaba las peticiones de clientes desde un equipo concreto, se decidió preparar una máquina virtual que alojara la aplicación. El Banco tenía un servicio de hosting de otra empresa en el que podían mantener varias máquinas, así que se comprobó con el Banco si podían incluir una más y resultó que sí podían incluirla sin aumentar el precio que ya pagaban. Joomla estaba programado en php y podía utilizar Javascript, un detalle a tener en cuenta para cuando se necesitase modificar el funcionamiento de algún componente de la aplicación. Por último,

se necesitaba un gestor de bases de datos para controlar la información que manejaría el sistema, así que se incluyó XAMPP en la aplicación.

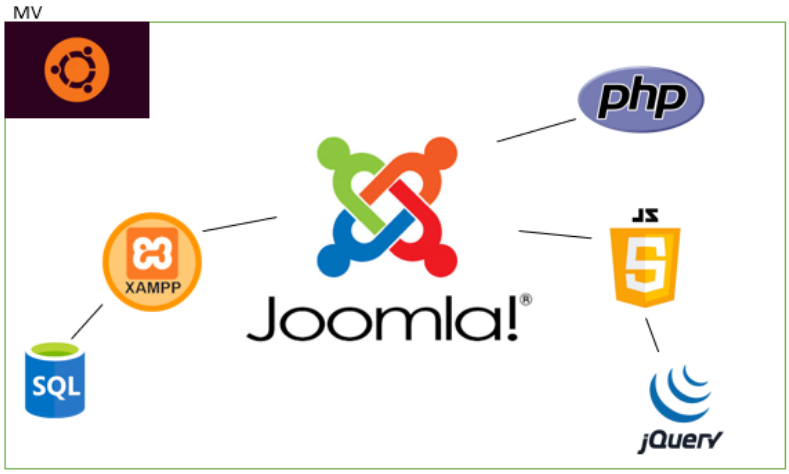


Figura 3.1: *Esquema de las herramientas utilizadas en el sistema*

Con la explicación en alto nivel de la arquitectura del sistema terminada, hay que explicar en detalle las herramientas que se han utilizado a lo largo del proyecto y porqué se escogieron.

3.2. Joomla

Es la herramienta más importante del proyecto, se utilizó para crear y gestionar la aplicación web. En la web oficial hay una tienda que ofrece muchas extensiones creadas por la comunidad, estas añaden funcionalidades de todo tipo a una página web que se esté desarrollando. Joomla y todas las extensiones gratuitas que están en la tienda oficial tienen una licencia GNU General Public License (GPL), lo que significa que se pueden descargar, usar, compartir y modificar sin ninguna restricción, incluso se puede vender un software que se haya diseñado con ellas, sea una extensión o una aplicación. En el desarrollo de SOL-BAM no se ha comprado ninguna extensión para implementar el proyecto. Se utilizó este CMS porque era un sofware sencillo de utilizar, era muy popular, luego si tenía algún problema podría encontrar alguna solución en internet fácilmente y porque era lo bastante flexible como para instalar extensiones que modificasen por completo la funcionalidad de la aplicación, luego tendría más opciones para implementar los objetivos.

Una aplicación de Joomla se organiza en 2 secciones, el frontend y el backend. El frontend es lo que suelen ver todos los usuarios, en él se muestran todas las funcionalidades a las que se quiere que acceda cualquiera en un uso normal de la aplicación, los administradores lo pueden modificar sin necesidad de reiniciar la máquina. El backend es la sección de la aplicación en la que se gestiona toda la información de la misma, en ella se pueden diseñar páginas a las que se quiere que se acceda

desde el frontend. Solo los administradores pueden acceder a esta sección.

Por defecto, no todos los administradores pueden instalar y configurar nuevo software, el único que puede es el super usuario, pero los permisos de cualquier usuario se pueden modificar según lo que se necesite.

Las páginas en Joomla están formadas por los siguientes elementos:

- **Menús y Elementos del menú**: Un menú es un conjunto de enlaces que apuntan a sus elementos. Un elemento del menú es una página web de la aplicación, hay de muchos tipos en función del elemento principal de la página, ya sea un artículo o un módulo concreto.

- **Artículos**: Son documentos escritos que pueden contener texto, imágenes, tablas, enlaces y mucho más. Sirven para mostrar información al usuario y que no tenga ninguna interacción con ella. Si se quiere implementar un blog en Joomla, es habitual incluir muchos artículos en la página. En SOL-BAM solo hay un artículo, está en la página de inicio.

- **Módulos**: Son elementos más complejos, en la mayoría de los casos ofrecen algún tipo de interacción con el usuario, como un calendario para gestionar eventos, un menú con enlaces a otras páginas o un formulario para registrarse. Cuando se instala una extensión que añade una funcionalidad para un usuario en el frontend, se está instalando un módulo nuevo. Una página no está limitada a un elemento, se pueden incluir varios módulos en páginas concretas.

Figura 3.2: *Página web con el artículo "Inicio", un módulo de búsqueda y un menú*

Para diseñar una aplicación con esta herramienta, se necesita un gestor de BBDD en tu equipo. Después hay que crear una carpeta en un directorio concreto del gestor, descargar Joomla y descomprimirlo en dicha carpeta. En XAMPP el directorio por defecto es "htdocs" en la raíz, en WampServer, es la carpeta "www".

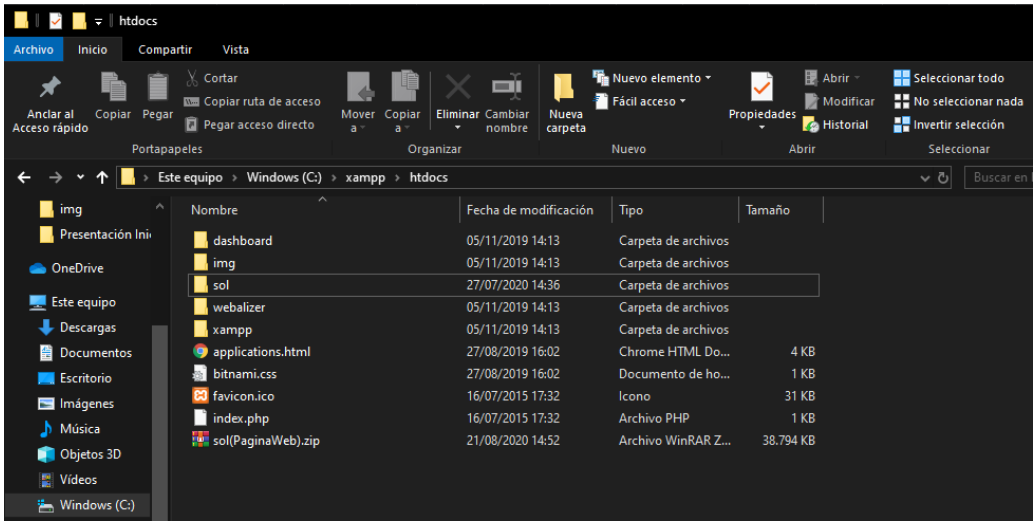


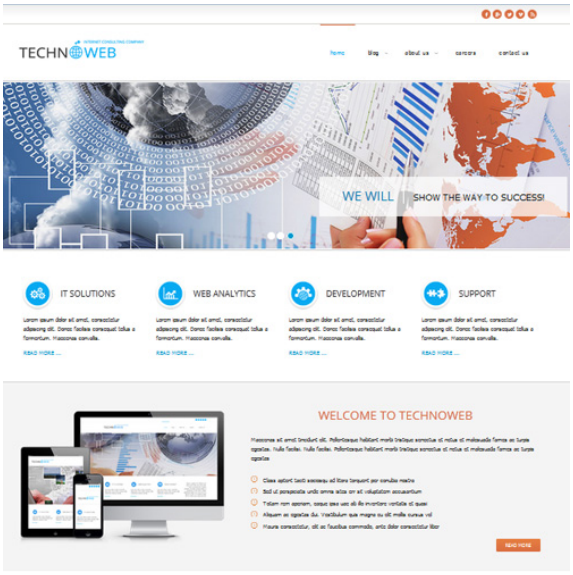
Figura 3.3: *La aplicación web "sol" en htdocs en XAMPP*

3.2.1. Extensiones de Joomla

Para implementar los objetivos del proyecto, se buscaron extensiones en la tienda oficial de Joomla durante el desarrollo. Antes de instalar una extensión se inspeccionaba la demo de la tienda, si tenía, y se revisaba su documentación para entender qué incluía y cómo usarlo, pero en algunos casos daban problemas en el equipo instalado (incompatibilidades con otras extensiones instaladas, errores de partes del código que no funcionaban, ...), en esos casos se desinstalaban y buscaban otras. En este apartado solo se menciona las que se utilizan en la versión final del proyecto.

- **La plantilla as002057**: Una plantilla de Joomla permite cambiar el aspecto visual de la página web en el frontend o el backend. Por defecto hay 2 plantillas para el frontend y 2 para el backend, pero se pueden descargar más. La plantilla as002057 modifica el frontend. Está diseñada con un framework de Bootstrap que permite estructurar los elementos de la página en la cabecera, el cuerpo o el pie de página. Además el cuerpo se estructura en 3 columnas, aunque la versión de pago tiene más.

  Se escogió esta plantilla porque tenía un estilo parecido al BAM, tonos azules sobre blancos, y permitía diseñar las páginas con comodidad gracias a la distribución de posiciones que ofrece.



(a) Captura de la demo oficial

(b) Ejemplo cabecera SOL-BAM

Figura 3.4: *Plantilla del proyecto*

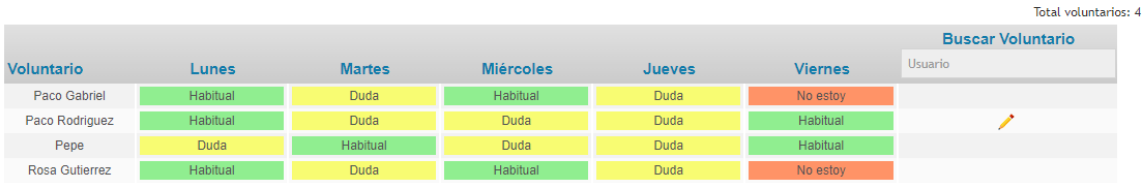
- **AJAX Seach**: Joomla ofrece un buscador por defecto para las páginas del proyecto. Esta extensión permite crear un buscador que muestra los resultados mientras el usuario está escribiendo en vez de cargar una página concreta con los resultados. Lo utilizan los usuarios en el frontend de la aplicación.

Figura 3.5: *Resultado de una búsqueda con AJAX Search*

- **CSSconfig**: Permite escribir instrucciones CSS que se cargan en todas las vistas de la aplicación. Simplifica crear o modificar el estilo de las páginas ya que todo ese código está en un mismo fichero. Se configura en el backend de la aplicación.

- **Custom Tables**: Crea tablas SQL que se guardan en la BD y las muestra en el frontend. Las tablas se pueden personalizar con muchos tipos de campos y se pueden hacer referencias entre sí. También se pueden crear plantillas que se asignan a vistas concretas, como ver toda la tabla o editar o añadir una fila.

  Hay que destacar que esta extensión es bastante compleja y la documentación de la versión gratuita es muy limitada.



Total voluntarios: 4

Buscar Voluntario

Usuario

Voluntario	Lunes	Martes	Miércoles	Jueves	Viernes
Paco Gabriel	Habitual	Duda	Habitual	Duda	No estoy
Paco Rodriguez	Habitual	Duda	Duda	Duda	Habitual
Pepe	Duda	Habitual	Duda	Duda	Habitual
Rosa Gutierrez	Habitual	Duda	Habitual	Duda	No estoy

(a) Vista de la tabla

```
<style>
.datagrid th{text-align:center;}
.datagrid td{text-align:center;}
</style>
<div style="float:right;">Total voluntarios: {recordcount:numberonly}</div>
<div style="float:left;" class="btnAnadirAsistencia">{add}</div>

<div class="datagrid">
{catalogtable:
"Voluntario":"<div class='nombreUsuario'> [usuario]</div>",
"*lunes*":"<div class='[lunes]'> [lunes] </div>",
"*martes*":"<div class='[martes]'> [martes] </div>",
"*miercoles*":"<div class='[miercoles]'> [miercoles] </div>",
"*jueves*":"<div class='[jueves]'> [jueves] </div>",
"*viernes*":"<div class='[viernes]'> [viernes] </div>",
"Buscar Voluntario<br/><div id="buscadorAsistenciaUsuario">{search:usuario}</div>":"<div class="botonesModificarAsistencia"><div
class='btnEditarAsistencia [usuario]'> {toolbar:edit} </div> <div class="btnEliminarAsistencia"> {toolbar:delete} </div> </div>";tablaAsistencias
}
</div>
<br/><div style='text-align:center;'>{pagination}</div>
```

(b) Plantilla de la vista

Figura 3.6: *Ejemplo tabla de Custom Tables*

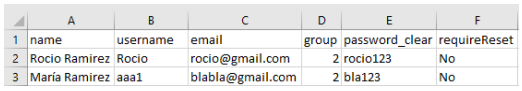
- **Issue Tracker**: Es un gestor de notificaciones. Está pensado para una página que gestiona varios proyectos. Permite enviar un aviso de una categoría concreta relacionado con un proyecto determinado para que los responsables de dicho proyecto estén informados y puedan actuar en consecuencia.

  También se pueden modificar los permisos de los usuarios de tal manera que se pueden generar responsables de varios niveles para cada proyecto.

Figura 3.7: *Notificaciones de usuarios*

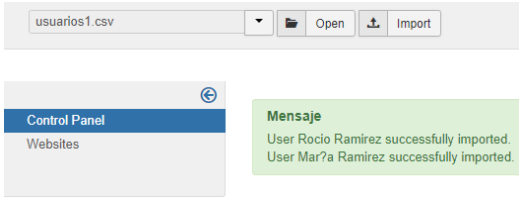
- **J2XML y Users Importer**: J2XML es un software que permite interpretar ficheros xml para que otras extensiones los lean y puedan ejecutarse. Users Importer importa los usuarios de un fichero xml, en el caso de que se produzca un error con un usuario concreto o que ya está en el sistema, se salta a la siguiente fila.

  Se escogieron porque son muy fáciles de utilizar y pueden agilizar el proceso de incluir los usuarios cuando se instale el sistema.



	A	B	C	D	E	F
1	name	username	email	group	password_clear	requireReset
2	Rocio Ramirez	Rocio	rocio@gmail.com	2	rocio123	No
3	María Ramirez	aaa1	blabla@gmail.com	2	bla123	No

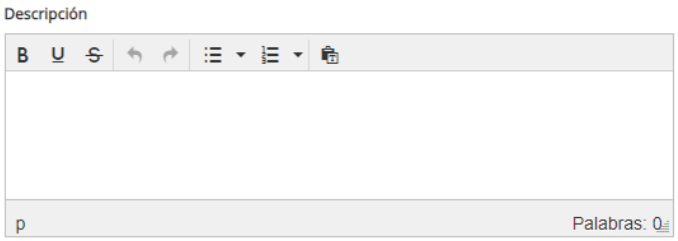
(a) Fichero Excel



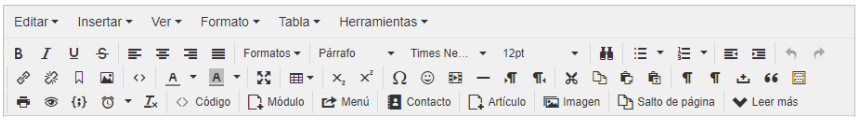
(b) Usuarios importados en la web

Figura 3.8: *Importando usuarios*

- **JCE**: Es un editor de texto que se puede utilizar en el backend y el frontend. Esta extensión permite a los usuarios escribir textos y complementarlos con listas, tablas, imágenes o enlaces a otras páginas. Se puede usar al crear un artículo, un módulo o al escribir cualquier campo de texto, como la descripción de una reserva en el frontend. Además, puede limitar los complementos a los que tenga acceso el usuario que lo esté utilizando en función de sus permisos.



(a) Editor JCE para un usuario normal



(b) Opciones del editor para un administrador

Figura 3.9: *Tipos de editores JCE*

- **JEvents**: JEvents es una extensión que gestiona eventos dentro de un calendario. Estos eventos pueden tener una o varias categorías, lo que permite clasificarlos y filtrarlos fácilmente. Además se pueden programar para que se repitan periódicamente.

  La versión gratuita incluye 5 plantillas para los calendarios y varias opciones de configuración como mostrar eventos específicos o limitar el acceso a usuarios. Hay varias versiones de pago y cada una aporta más o menos plugins para facilitar su uso.

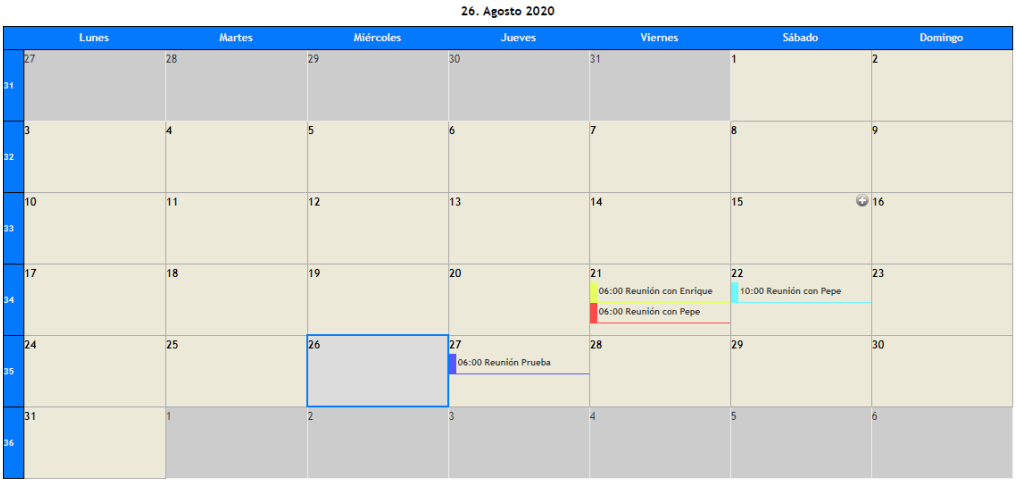


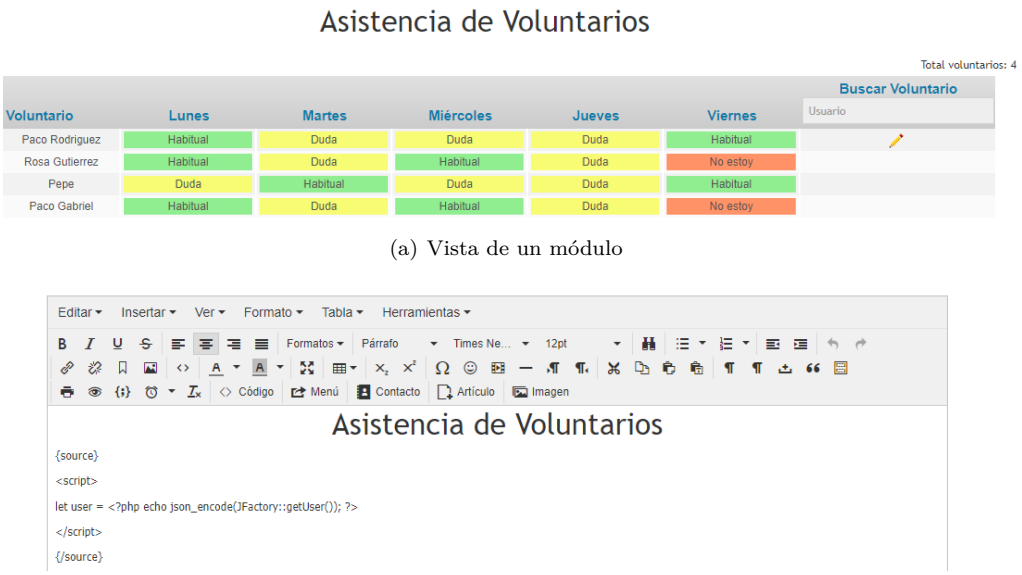
Figura 3.10: *Calendario con eventos*

- **JQuery Easy**: Es una herramienta muy potente que permite modificar el funcionamiento de Javascript y sustituir unos frameworks por otros. Los frameworks que controla son MooTools,

JQuery y la biblioteca Bootstrap. Esta extensión es muy popular en aplicaciones que tienen problemas de conflictos entre MooTools y JQuery.

También puede importar scripts externos o escribirlos en un editor que incorpora la herramienta, en ambos casos se incluyen en todas las páginas. Toda esta gestión se hace en el backend.

- **Sourcerer**: Permite incluir código php o Javascript en artículos y módulos. Se utilizó en la aplicación para escribir pequeños script php que solo iban a ser necesarios en unas páginas concretas.



(a) Vista de un módulo

(b) Módulo en el editor del backend

Figura 3.11: *Sourcerer en un módulo*

- **uddeIM**: Es un Private Message System (PMS), ofrece a los usuarios la posibilidad de ponerse en contacto entre ellos sin utilizar herramientas externas como Whatsapp o que se tengan que levantar de su puesto. Tiene muchas similitudes con un sistema de emails como Gmail.

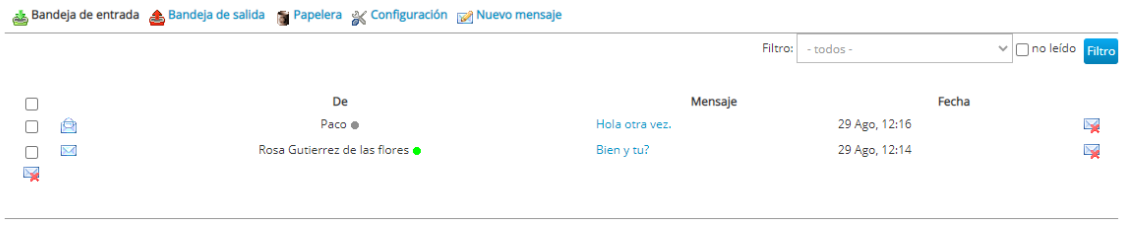


Figura 3.12: *Bandeja de entrada uddeIM*

3.3. Lenguajes de programación

Joomla y cualquiera de sus extensiones y plantillas está desarrollados con php, sin embargo puede interpretar scripts creados con Javascript. Joomla utiliza Javascript dentro de un código php y no le permite acceder a algunos datos. Un problema que surgió durante el desarrollo fue acceder al usuario de la sesión con Javascript, para hacerlo fue necesario crear una variable en php con los datos que se necesitaban y buscarla desde el primer script. Durante el desarrollo no se tuvo que programar mucho, solo se hizo para modificar algunos elementos cuando no estaba la opción de editarlos desde la extensión que los controlaba.

Cuando se empezó el desarrollo el autor del trabajo no tenía ninguna experiencia con php, así que se decidió evitarlo si había otra opción. Se programó todo lo que necesitaba en Javascript, pero hubo un problema importante al principio, Joomla utiliza 2 frameworks que provocan conflictos, Mootools, que está orientado a la programación de objetos y actuar independientemente del navegador que use el cliente, y JQuery que permite acceder a elementos de ficheros html y modificar su comportamiento. Estos conflictos se producen porque ambos frameworks acceden a los mismos recursos y se les puede llamar a la vez. Para solucionar este problema, se instaló la extensión JQuery Easy que permitió generar los scripts que se necesitaban y evitar las llamadas a Mootools cuando no lo necesitaba la página para evitar conflictos.

3.4. XAMPP y bases de datos SQL

Como gestor de BBDD se decidió instalar XAMPP porque se utilizó en la carrera y no causó ningún problema en ningún trabajo previo. Para encender la aplicación web solo se necesitaba activar el módulo de Apache, que escuchaba los puertos 80 y 443, y el MySQL, puerto 3306. XAMPP permite gestionar los datos del sistema con phpMyAdmin.

Joomla utiliza una BD llamada “sol-bam”, en la que genera varias tablas, todas ellas empiezan su nombre con “qjapr” para que no acceda a otras tablas que no se quieran usar en la página web. Estos 2 nombres se pueden cambiar en la configuración global de Joomla. No es necesario gestionar manualmente la BD en un uso normal ya que Joomla la controla automáticamente.

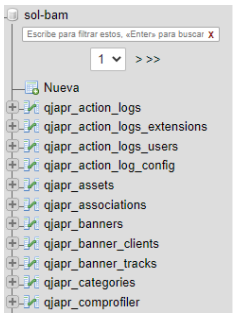


Figura 3.13: *Captura de la BBDD en phpMyAdmin*

3.5. Máquina virtual

Antes de crear una máquina se contrastó las ventajas y los inconvenientes de varios sistemas operativos para escoger el más efectivo para el proyecto. Al final se decidió utilizar Ubuntu Server en la máquina porque estaba orientado a escuchar peticiones de otros equipos, era fácil de utilizar para alguien que no tenía experiencia con un sistema Linux, se podía instalar y utilizar sin pagar nada y era muy popular, por lo tanto si había algún problema se podía buscar una solución por internet rápidamente.

Mientras se buscaba el sistema para descargarlo se encontró Osboxes, una web en la que subían imágenes de máquinas virtuales con sistemas Linux instalados para que cualquiera las utilizase en lo que las necesitara. Se buscó una máquina con Ubuntu Server, se descargó y se importó en el gestor de máquinas virtuales.

El gestor que se utilizó fue VirtualBox porque se había trabajado con él y las imágenes de Osboxes estaban configuradas para funcionar con este software, aunque luego se pudieran exportar a un formato que pudieran utilizar otros gestores.

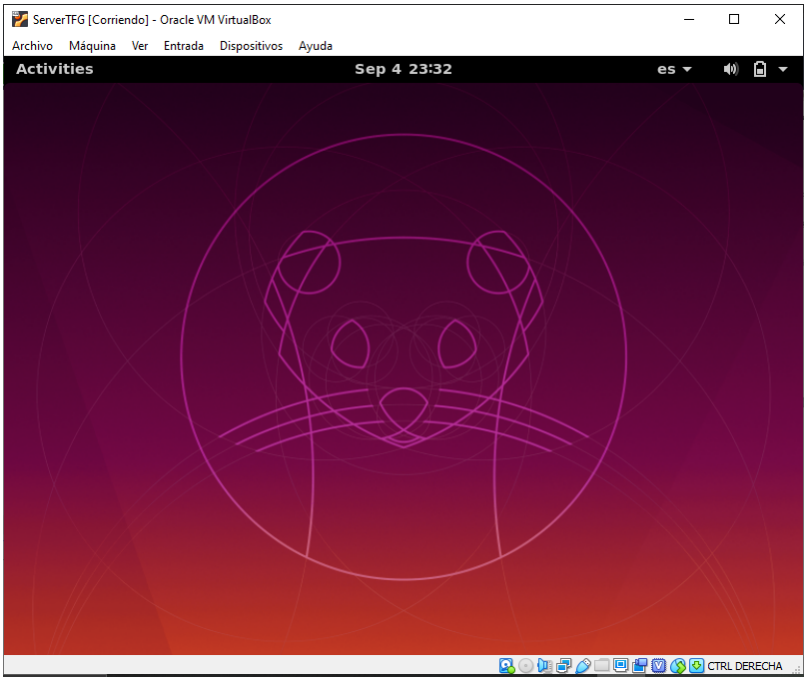


Figura 3.14: *Máquina virtual en VirtualBox*

Capítulo 4

Desarrollo

Una vez que se terminó de investigar qué herramientas se iban a utilizar, se empezó el desarrollo del proyecto. En este capítulo se va a explicar en detalle el proceso del desarrollo del sistema.

Primero se implementó la aplicación web ya que era el proceso más importante y el que llevaría más tiempo para terminarlo. Se desarrollaron los requisitos funcionales de uno en uno, de tal manera que no se avanzaba al siguiente hasta que no se terminaba el actual.

Con la aplicación terminada y funcional, se preparó la máquina virtual y se trasladó todo el software a ella, concluyendo con la implementación del proyecto. Para confirmar el correcto funcionamiento de la aplicación, se realizaron varias pruebas del sistema que se explicarán al final del capítulo.

4.1. Inicio y gestión de usuarios

Al empezar el desarrollo se buscó la versión segura más avanzada de Joomla y se instaló en el equipo. Durante la instalación se escogió la opción de generar unas páginas por defecto, así había un ejemplo completo y funcional en la aplicación que permitiría entender el funcionamiento de la herramienta en detalle. También se buscaron tutoriales en blogs de internet y pequeños cursos en Youtube.

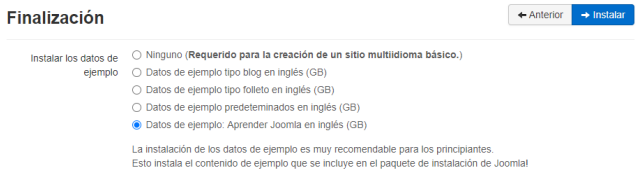


Figura 4.1: *Opciones de datos de ejemplo en la instalación*

El sistema con la opción escogida contenía 2 páginas web de ejemplo, la primera era un blog con una galería de fotos bastante simple, mostraba cómo funcionaban los artículos con varias configuraciones y algún módulo concreto como las galerías, y la segunda simulaba una tienda online de fruta. Este último era más complejo y tenía varios apartados con módulos de todo tipo, además de una plantilla diferente para cambiar el frontend de la página. Cuando se entendió el funcionamiento interno de la herramienta, se borraron los ejemplos y se empezó a desarrollar la aplicación.

Para empezar, se decidió implementar los usuarios y los roles que se iban a utilizar. Los roles que se necesitaban en la aplicación eran "administradores‘" para controlarlo todo, "usuarios normales" que tenían acceso a todas las funcionalidades de la aplicación y los "informáticos" que podían gestionar las notificaciones de problemas de usuarios normales. En Joomla los usuarios no tienen roles como tal, forman parte de uno o varios grupos de usuarios a la vez con más o menos permisos. Para los 2 primeros roles se decidió utilizar los grupos "Administrator" y "Registered" que vienen por defecto en el sistema, pero para los informáticos se creó un nuevo grupo, "informatico", que tenía los mismos permisos que un usuario normal. Más adelante se revisaría este rol y se modificaría en función de lo que se necesitaría para las notificaciones. Se dejaron los demás grupos que estaban por defecto en el sistema por si en el futuro el Banco los fuera a necesitar.

Figura 4.2: *Grupos de usuarios*

Mientras se hacían pruebas creando usuarios y asignándoles grupos se detectó que, cuando se quisieran incluir muchos usuarios a la vez, por ejemplo cuando se esté instalando el sistema por primera vez, el proceso iba a ser muy pesado, así que se optó por buscar alguna forma de incluir muchos usuarios desde un fichero externo. Se encontraron muchas extensiones, pero se decidió utilizar J2XML y User Importer. Después de instalarlo y probarlo, se empezó a trabajar en el frontend de la aplicación.

Antes de empezar con los objetivos, se necesitaba buscar qué plantilla se iba a utilizar para el frontend de la aplicación, así se podría preparar la vista final de cada funcionalidad mientras se terminaban durante el desarrollo. Mientras se revisaban las demos de las plantillas que se consideraron interesantes, no pasó desapercibido un pequeño buscador que incluían algunas demos para encontrar páginas concretas. Se consideró una funcionalidad extra muy interesante para el proyecto, así que optó por incorporarla después de escoger la plantilla.

Con la plantilla instalada y las pruebas de compatibilidad con el sistema previo terminadas, se probó el buscador por defecto de Joomla, pero acabó siendo de poco valor porque mostraba los resultados en una página nueva, por lo que se buscó alguna extensión que permitiera incluir un buscador como el las demos visitadas. Cuando se encontró una extensión con un buscador así, AJAX Search, se instaló y se confirmó que no provocaba conflictos con ningún componente previo. Después de configurar todo se decidió crear algunos artículos de ejemplo y un menú para preparar una pequeña demo de la aplicación. Con la demo terminada, se volvió a contactar con el Banco para tener una reunión y enseñar los avances.

Esta reunión se realizó en julio del 2020 y en ella se mostró la demo y los objetivos que se iban a abarcar para que confirmaran que seguían siendo de interés para el Banco. Al responsable del Banco le gustó la presentación del software, confirmó las funcionalidades que estaban recogidas y pidió añadir otra más, el sistema para controlar las visitas del Banco. Indicó que esta funcionalidad era muy concreta y que se iba a usar muchas menos veces que el resto, pero que les vendría bien un sistema que controlara esa parte. Se explicó que se podía incorporar como el requisito funcional menos importante, pero no había certeza de poder implementarlo a tiempo. El responsable estuvo de acuerdo y dejó claro que si se podía hacer perfecto, pero que preferían que el resto de objetivos de la lista estuvieran implementados antes. Con la reunión terminada, se continuó desarrollando la aplicación.

4.2. Asistencia de voluntarios

Este apartado del proyecto resultó ser el más complicado del desarrollo porque se diseñaron varias ideas para implementar una solución al problema y se tardó mucho tiempo en encontrar una opción viable.

Para conocer la asistencia de cada usuario, se pensó en que cada persona tendría dos tablas, una que representaba que días de la semana solía estar y la otra que indicaba los meses. Ambas tablas debían ser modificables para actualizar la información cuando el usuario quisiera y cada campo debía tener 1 valor de 3, “Habitual” que indicaba que era raro que el usuario faltara en ese momento, “Duda”, que no siempre estaba, y “No estoy”. Cualquiera podía indicar que, por ejemplo, no estaba en agosto ningún día pero que el resto del año solía estar los viernes. Con esta estructura se podía guardar mucha información de los voluntarios sin ser muy invasivo, y si alguien quería actualizar su asistencia lo podía hacer rápidamente.

Se optó por guardar estas tablas en los perfiles de cada usuario y que con el buscador se pudieran encontrar los perfiles de los demás, pero se produjeron 2 problemas graves:

- Se encontraron muchas extensiones que permitían añadir campos al perfil de los usuarios, pero ninguna incluía una tabla con filas y columnas. Se podía solucionar incluyendo muchos campos de texto o una opción similar, pero los perfiles serían muy grandes y lo convertirían en un sistema muy pesado de utilizar.
- Los perfiles de los usuarios se trataban por defecto como información privada y solo se podía acceder al perfil del usuario conectado. No se encontró ninguna extensión que permitiera hacer públicos los perfiles del sistema.

Se encontró una extensión que duplicaba los perfiles de Joomla de tal manera que se podían tratar como si fuera una red social parecida a Facebook o Twitter. Esta extensión era Community Builder de Joomlapolis y ofrecía a los usuarios la posibilidad de crear posts para que otros los vieran y escribieran comentarios, aparte de que las cuentas se podían hacer públicas. Si bien es cierto que esta extensión solucionaba uno de los problemas, también generaba otros.

Con esta extensión no se podía generar tablas en los perfiles, solo se podían generar en un post, por lo tanto los administradores debían confirmar que cada cuenta solo tenía posts con dichas tablas. Además no había una forma clara de limitar los comentarios en la aplicación y no se podía enviar mensajes privados entre usuarios. Por último, la extensión daba conflictos con la plantilla que se estaba utilizando. Se podía cambiar la plantilla por otra, pero nada aseguraba que otra plantilla que no estuviera diseñada por Joomlapolis funcionase, aparte de que aparentaba ser un software muy complejo de manejar y llevaría mucho tiempo configurarlo para hacer lo que se necesitaba. Al final se descartó la opción de los perfiles para guardar las tablas.

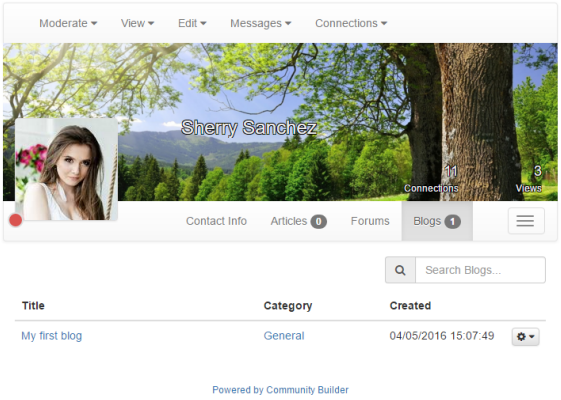


Figura 4.3: *Perfil de ejemplo de Joomlapolis*

La siguiente opción que se consideró fue generar una página concreta para cada usuario de tal manera que solo el dueño podría editarla. En esta ocasión no se encontró un plugin que permitiera hacer algo así, la única opción que había era diseñar la extensión manualmente. Antes de descartar esta opción se buscó como se podía hacer una extensión en internet y no era un proceso tan sencillo como se esperaba. Una extensión de Joomla debía seguir una estructura de ficheros y directorios concreta para que el sistema lo reconociera como un software que podía ejecutar. Aparte del tiempo

que llevaría hacer una extensión, se tendría que controlar como podría modificar un usuario su página, ya que, de no hacerlo, podría haber cualquier cosa en la página en vez de las tablas. Por estos motivos, se descartó esta opción y se empezó a buscar otra.

La última posibilidad que surgió fue diseñar 2 páginas, cada una con una tabla con la asistencia de los perfiles del sistema, todos los usuarios accederían a las mismas páginas y solo podrían modificar la fila de su perfil. Mientras se buscaba alguna extensión qué gestionara tablas se encontró Custom Tables, un plugin que permitía diseñar y controlar tablas de datos. Cuando se estuvo buscando documentación de esta herramienta no se encontró mucho, pero lo que se vio en la demostración era muy interesante y los otros gestores de tablas que se revisaron en la tienda oficial tenían menos opciones de edición en las vistas, así que se optó por esta extensión.

Se empezó diseñando la tabla que guardaba los datos de la semana, la estructura de todas las tablas iba a ser casi igual salvando que haya más o menos campos. Todas las filas iban a tener un usuario, sería una clave foránea que apuntaría a los usuarios del sistema y la clave principal de la tabla, por lo que no se podría hacer más de una fila que apuntase al mismo usuario. el resto serían campos de texto, pero se tendría que limitar sus posibles valores de alguna forma para que solo pudieran ser “Habitual”, “Duda” y “No estoy”.

Nombre del campo	Título del campo	Tipo de campo	Parámetros de tipo	Necesario	Tabla	Estado	Carné de identidad
lunes	• English (en-GB): **Monday** • Spanish (español): **Lunes**	Cadena de texto (línea única)	255	No	Week Schedule	✓	1
martes	• English (en-GB): **Tuesday** • Spanish (español): **Martes**	Cadena de texto (línea única)	255	No	Week Schedule	✓	2
miercoles	• English (en-GB): **Wednesday** • Spanish (español): **Miércoles**	Cadena de texto (línea única)	255	No	Week Schedule	✓	3
jueves	• English (en-GB): **Thursday** • Spanish (español): **Jueves**	Cadena de texto (línea única)	255	No	Week Schedule	✓	4
viernes	• English (en-GB): **Friday** • Spanish (español): **Viernes**	Cadena de texto (línea única)	255	No	Week Schedule	✓	5
usuario	• English (en-GB): **User** • Spanish (español): **Usuario**	Usuario	,,,,unique	Sí	Week Schedule	✓	6

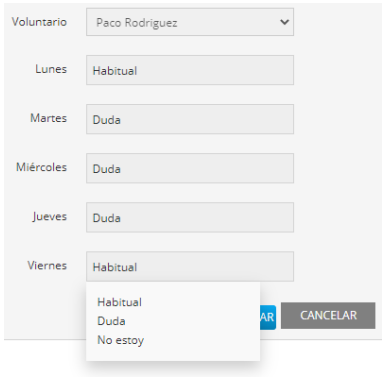
Figura 4.4: *Campos de la tabla Semana*

Gran parte de lo que se hizo en las tablas fue con las opciones que daba la extensión, pero se necesitaba cambiar algunos detalles para que funcionara como se necesitaba, así que se decidió buscar la forma de introducir scripts de Javascript para modificar el funcionamiento de algunos elementos y un fichero CSS para personalizar la vista. Como no había ninguna forma de introducir algo así salvo modificando la extensión, se decidió buscar otras extensiones que permitieran editar e incluir lo que se necesitaba.

Se utilizó la extensión CSSconfig para modificar algunos elementos de las vistas de la tabla con un editor CSS que incluía la herramienta, por ejemplo, cambiar el color de los campos de las filas. Con la extensión JQuery Easy se pudo introducir algunos scripts Javascript para limitar las opciones de los campos de texto, generando un contenedor que sólo era visible cuando se ponía el

ratón encima del campo que se quería modificar.

(a) Vista de la tabla modificada con CSSconfig siendo un administrador



(b) Opciones del contenido con JQuery Easy

Figura 4.5: *Modificaciones con CSSconfig y JQuery Easy*

Para terminar la tabla de la semana, se tenía que configurar para que el usuario solo pudiera modificar su fila, salvo si era un administrador. Cuando se empezó a implementar esa parte, se descubrió que no se podía acceder a la sesión desde un script en Javascript, por lo que se requería algún otro componente que accediera a los datos que se necesitaban. Se buscó alguna extensión que permitiera generar scripts en php y se encontró el plugin Sourcerer. Con esta extensión se podía introducir código en otros módulos, así que se creó un script que accediera a la sesión y guardara la información que se necesitaba en una variable. El script Javascript leería esa variable y reconocería qué fila era el usuario conectado, impidiendo que se pudieran editar otras filas.

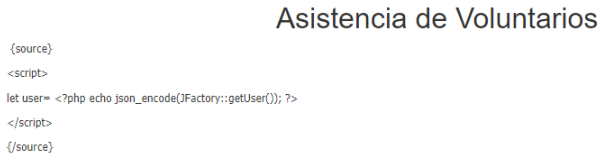


Figura 4.6: *Script generado en un módulo con Sourcerer*

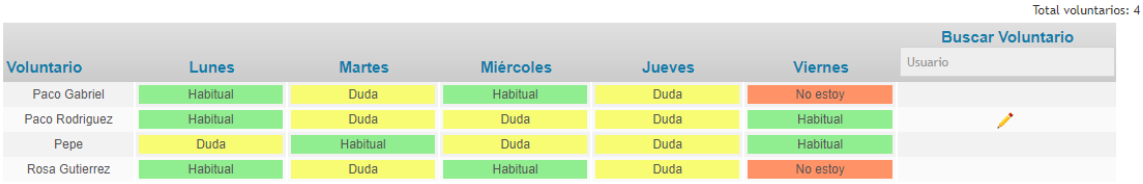
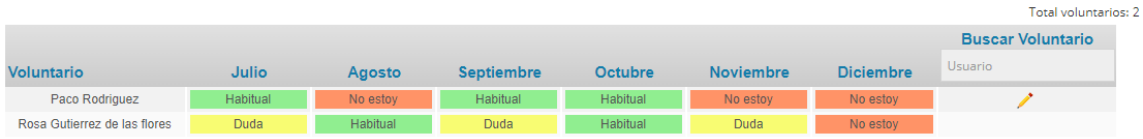


Figura 4.7: *El usuario solo puede modificar su fila*

Cuando se terminó con la tabla de asistencia de la semana, se empezó a diseñar la de los meses. Dado el resultado en pantalla, se decidió separarla en 2 semestres, así se leería cómodamente el contenido de las celdas. Se importaron los mismos scripts que había antes y se generaron las mismas plantillas con los campos de los meses. Lo único que se cambió respecto a la tabla de la semana es adonde apuntaban los usuarios, apuntaban a los de la semana. Esto se hizo así porque en el Banco estaban más interesados en la asistencia semanal, no se iba a dar el caso de un usuario que tuviera las asistencias de cada mes y no las de cada día. Otro punto a tener en cuenta es que el borrado de las filas en estas tablas es en cascada, por lo tanto si un voluntario dejaba de trabajar en el Banco, solo se tendría que borrar su fila de la semana.

(a) Tabla primer semestre



(b) Tabla segundo semestre

Figura 4.8: *Tablas de asistencia de los meses*

Al implementar la asistencia en unas tablas generales a las que todos tenían acceso, cualquier miembro del Banco podía comprobar fácilmente qué compañeros iban a ir un día concreto, lo que simplificaba la planificación de cualquier tarea y agilizaba el proceso interno de la Fundación.

4.3. Sistema de reservas de salas

Con la gestión de la asistencia de voluntarios del Banco terminada, se empezó a diseñar el sistema para reservar salas de reuniones.

Al principio se pensó en utilizar alguna extensión qué permitiera hacer reservas parecidas a las que se harían en una biblioteca, un hotel o un alquiler de cualquier tipo. Estos sistemas permitían que los clientes vieran qué productos tenía la empresa y en qué fechas estaban disponibles, después se podían reservar para que nadie pudiera acceder a ellos mientras los estaban utilizando. Cuando se buscó un software así, se descubrió que Joomla tenía muchas extensiones de este tipo, reservas de hoteles como Jomres, gestores de rentas de propiedades como Rentalot o préstamos de contenido multimedia como MediaLibrary Basic. Si bien era bueno que hubieran muchas extensiones entre las que elegir, había un problema, todas ellas estaban centradas en un ámbito específico y, aunque se podían usar para otros contextos, habría que trabajar de más para ajustar el software y que funcionase en el proyecto. Un ejemplo eran los sistemas de los hoteles, en los que había un proceso por detrás para gestionar los pagos que no era necesario en el proyecto, aparte del hecho de que no se podía reservar una habitación de un hotel para 1 hora.

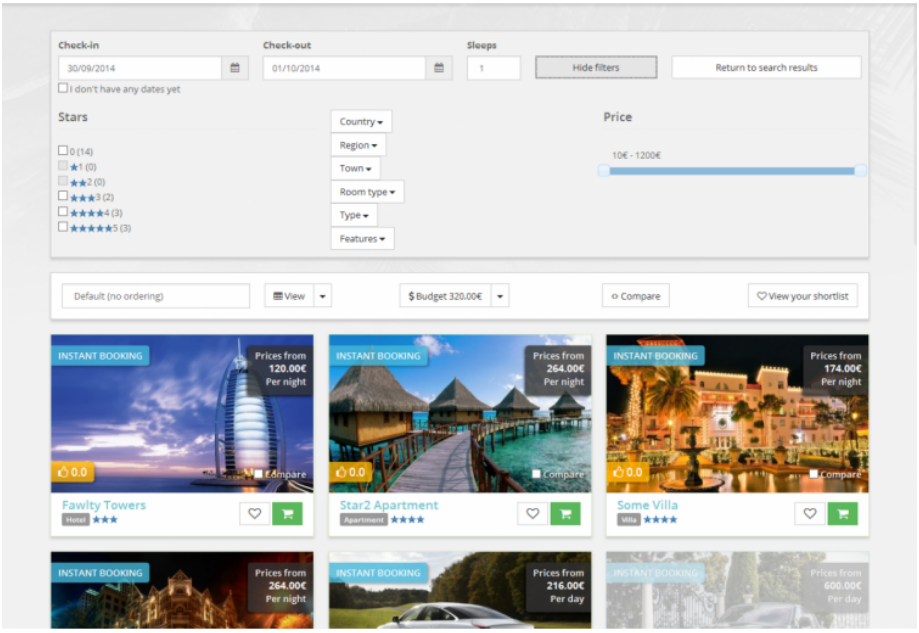
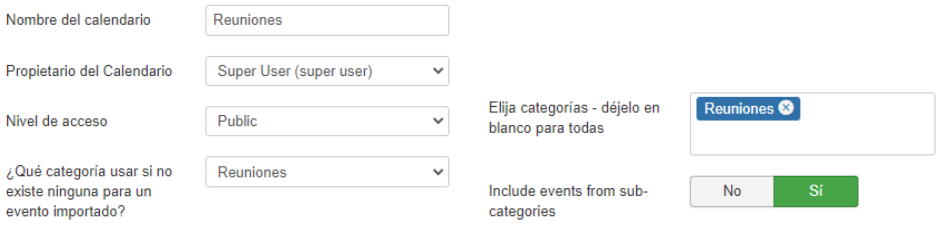


Figura 4.9: *Captura demo Jomres*

Debido a la especificación de estas herramientas, se optó por buscar una extensión más genérica que gestionara cualquier tipo de evento. Con un software así se tratarían las reuniones como eventos concretos que todos podrían ver desde la página web, pero que solo el creador de la reunión podría modificar o eliminar.

Se encontró una extensión, JEvents, que organizaba eventos en un calendario. Estos eventos podían tener una categoría para distinguirlos de otros. Además, la documentación era muy extensa y había acceso a todas las opciones de configuración en la versión gratuita, así que se instaló en el proyecto.

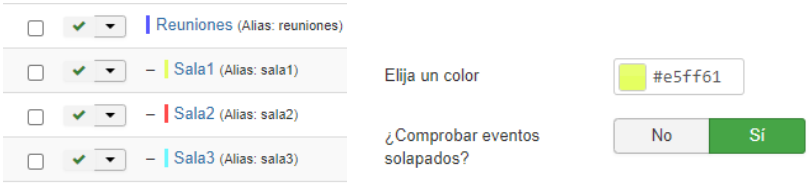
Antes de crear eventos, se tenía que construir un calendario en el que se guardarían. Mientras se revisaban las configuraciones, se encontró la posibilidad de limitar que tipos de eventos se podían generar en cada calendario. Con esta opción, se podía utilizar esta extensión en futuras implementaciones de la aplicación, lo que aumentaba el valor del proyecto. Se creó la categoría "reuniones" y varias subcategorías de ejemplo que representaban las salas que se podían reservar. Esto se diseñó así para que, si un usuario no tuviera claro que sala escoger, se le podría asignar a la reunión la categoría general y, más adelante, modificarla a una sala disponible.



(a) Configuración general (b) Categorías permitidas

Figura 4.10: *Configuración de categorías en el calendario*

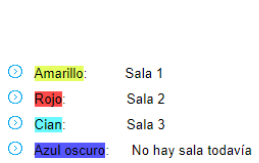
En las subcategorías se activó una opción que revisaba que no hubiera conflictos de horarios de reuniones con la misma categoría, así no habrían varias reservas de la misma sala a la misma hora.



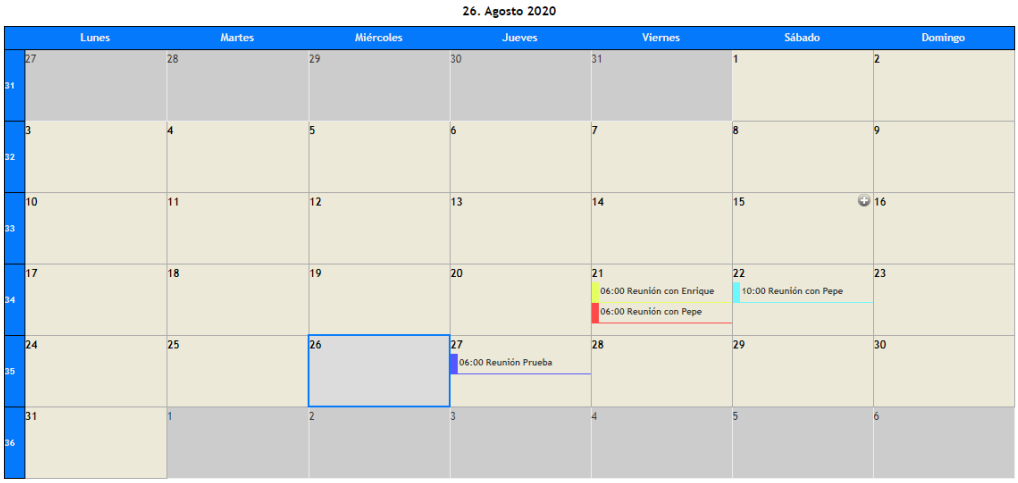
(a) Categorías de las reservas (b) Opciones subcategorías

Figura 4.11: *Subcategorías de las reservas*

Para terminar, se revisaron las plantillas por defecto de los calendarios, se escogió la más interesante y se ajustó con CSSconfig para que mantuviera los tonos que se utilizaban en el resto de la aplicación. También se añadió un título a la página con una leyenda para que los usuarios pudieran reconocer las reuniones por los colores asignados.



(a) Leyenda con las salas



(b) Calendario en el frontend

Figura 4.12: *Subcategorías de las reservas*

Al terminar esta parte del desarrollo, el proyecto tenía implementado un gestor de salas de reuniones que se podía ajustar al Banco y que se podía modificar en el futuro añadiendo o quitando salas al calendario. También tenía instalada una extensión que podía generar más calendarios y que trataría la información de cada uno por separado, asegurando que no se cruce información entre ellos.

4.4. Mensajería entre usuarios

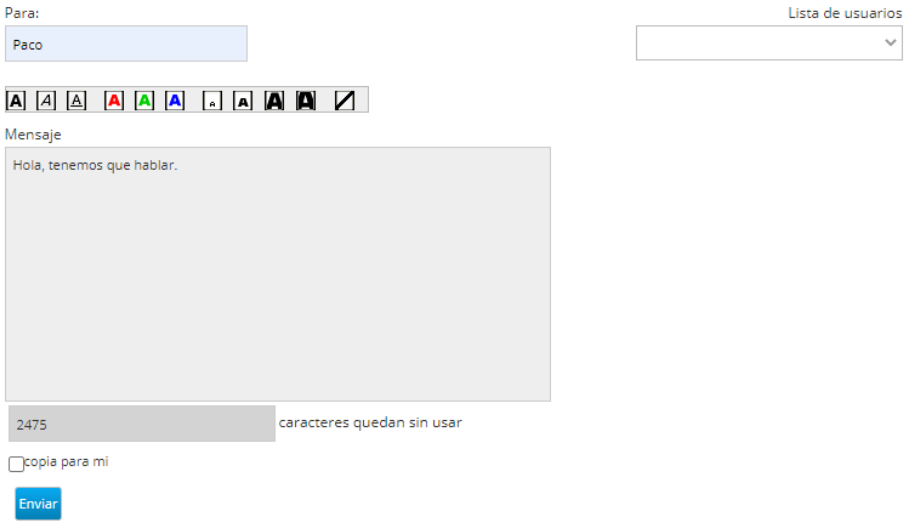
En este momento del desarrollo se implementó el tercer objetivo del proyecto, un sistema que permitiera a los usuarios comunicarse entre ellos.

Cuando se investigó qué extensiones permitían enviar mensajes entre usuarios se encontraron muy pocas opciones, el único tipo de sistema disponible era un PMS (Private Message System) y, de estos, solo había 3 en la tienda oficial que fueran gratuitos. Se escogió UddeIM porque era la extensión más valorada y con la mejor puntuación, las otras 2 tenían una valoración muy inferior y

no inspiraban ninguna confianza.

UddeIM era muy parecido a un servicio de correos electrónicos como Gmail o Outlook, cada usuario tenía una bandeja de entrada con los mensajes recibidos, otra de salida con los enviados y la opción de escribir un nuevo mensaje a uno o varios usuarios. Cuando se escribía un mensaje, estaba la opción de buscar entre todos los usuarios a los que se les quería enviar dicho mensaje, sin embargo, esta lista se podía modificar para que algunos usuarios no estuvieran. Esto permitía que hubiera administradores u otros usuarios específicos que no podía ver nadie en el sistema, como un súper usuario.

(a) ID's de los usuarios invisibles



(b) Escribir un nuevo mensaje

Figura 4.13: *Nuevo mensaje y opciones de usuarios*

Cuando se recibía un mensaje se podía responder sin necesidad de entrar en la opción de enviar un mensaje, así se podían iniciar chats entre los usuarios, igual que al responder un correo en otros servicios similares. También se podía ver si un usuario estaba conectado en la aplicación desde cualquier bandeja con un círculo verde al lado del nombre, por si se prefería responder al mensaje cuando su destinatario estuviera conectado.

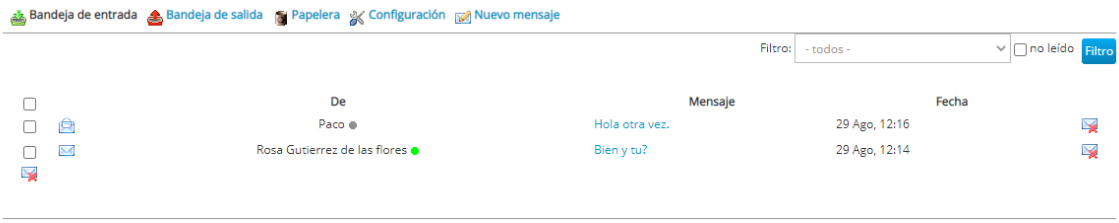


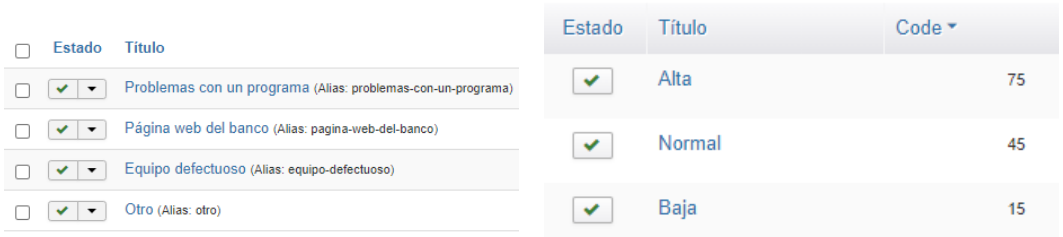
Figura 4.14: *Bandeja de entrada con 2 mensajes*

Con esta implementación terminada, los usuarios tenían una herramienta para comunicarse entre ellos en el momento, cumpliendo el tercer requisito funcional del sistema.

4.5. Notificación de problemas

Con el sistema de comunicación implementado y funcional, se tenía que desarrollar el cuarto objetivo y requisito funcional, un gestor de notificaciones que permitiera a cualquier usuario enviar avisos de problemas a los técnicos del departamento de Informática del Banco.

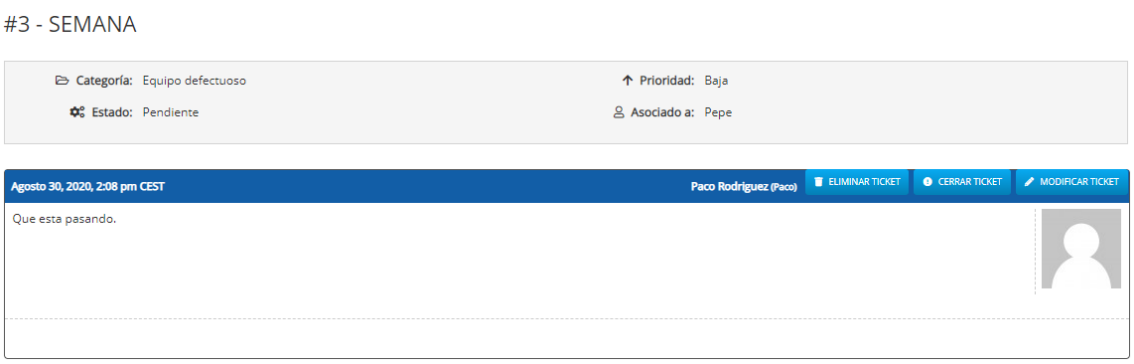
Al principio del desarrollo se instaló una extensión diferente a la de la versión final, JV-HelpDesk. Esta extensión permitía a los usuarios crear tickets para los administradores o cualquier otro usuario que los gestionase. Se podían generar categorías para estructurar las notificaciones en tipos y se podían especificar varios niveles de prioridad, cada uno tenía un valor numérico del 0 al 100 siendo el 100 el más importante. Las notificaciones también tenían un estado, este cambiaba automáticamente en función de lo que hicieran los administradores. Un ticket sin revisar estaba en "Abierto" para que cualquier administrador lo viera, cuando un responsable lo leía y se hacía cargo de él, se lo asignaba a sí mismo pasando al estado "Pendiente" y solo el responsable en cuestión tenía acceso a él. Cuando un responsable resolvía el problema, podía cerrarlo, pasando al estado "Cerrado".



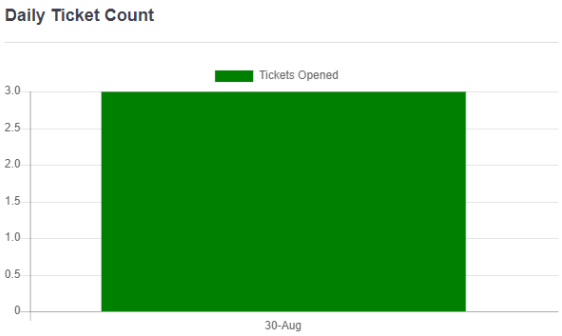
(a) Categorías (b) Prioridades

Figura 4.15: *Categorías y prioridades de los tickets en JV-HelpDesk*

Cuando un usuario generaba un ticket le asociaba una categoría y una prioridad y se le asignaba automáticamente el estado “Abierto”. La extensión recogía todos los tickets para generar gráficos que se podían ver en el backend. Para generarlos se tenía en cuenta la categoría, el valor numérico de la prioridad y su estado. Estas gráficas mostraban información de todo tipo y permitían a los responsables revisar el estado actual de las notificaciones en el sistema, de esta manera se podían organizar para abarcar todos los problemas eficientemente.



(a) Ejemplo notificación “JV-HelpDesk”



(b) Gráfico simple de problemas creados

Figura 4.16: *Notificación y gráfico de notificaciones*

La herramienta también permitía generar comentarios en los tickets, solo podían hacerlo el responsable asignado y el usuario que lo creó, de tal manera que se podían comunicar para intentar arreglarlo. Sin embargo, la herramienta empezó a dar problemas cuando se implementaron los comentarios. Algunos no se guardaban en el sistema o incluso no se guardaba quién lo había escrito. Si bien es cierto que algunos errores se pudieron corregir, la mayoría no, así que no se podía utilizar esa funcionalidad, y cuando se buscó la forma de desactivarla no se encontró ninguna. Esto implicaba que, si se dejaba esta herramienta en el software, se dejaría un componente que no funcionaba correctamente. Esto provocó que se buscara otra extensión.

Para evitar esta situación otra vez, se decidió buscar una extensión más simple con una documen-

tación que explicase detalladamente como funcionaba la herramienta y que tuviera más opiniones en la tienda oficial, para asegurar su funcionamiento. De las extensiones disponibles había una que cumplía todas las condiciones anteriores, Issue Tracker, que tenía un funcionamiento muy parecido a la anterior herramienta. Issue Tracker estaba pensado para una empresa que gestionaba varios proyectos de cualquier tipo. Cuando se enviaba una notificación se debía indicar de qué proyecto era. En el sistema se creó un proyecto, "Problemas", y cualquier notificación que se crease se le asignaba por defecto este proyecto, de tal manera que se hacía esta gestión internamente, permitiendo a los usuarios centrarse en campos más importantes.

Cuando se creaba una notificación también se tenía que seleccionar un tipo y una prioridad. Los tipos eran idénticos a las categorías de JV-HelpDesk, y las prioridades tampoco cambiaron. En esta extensión no había ninguna gráfica para evaluar la situación general. En el caso de los estados, Issue Tracker también permitía asignarle uno a la notificación, pero estos no cambiaban automáticamente. De los estados había uno que era el que se asignaba por defecto al crearse, "Abierto" y el otro que era un estado final, "Cerrado", cuando se asignaba no se podía volver a modificar ningún campo, pero seguía estando en el sistema para acceder a él en el futuro.

(a) Tipos de notificaciones (b) Estados (c) Prioridades

Figura 4.17: *Campos generados para clasificar las notificaciones en la versión final*

Esta extensión tenía muchos más campos que añadían información, como la localización donde se produjo la incidencia, pero había uno muy importante, la solución que se había encontrado para el problema. Era importante porque en ella se podía guardar una descripción detallada de como se resolvió el problema, de tal manera que, en el futuro, si surgía una situación similar, se podía repetir la solución.

Se configuró el grupo de usuarios "informático" para que pudieran gestionar las notificaciones igual que lo haría un administrador. Además, cada usuario solo podía ver sus notificaciones mientras que los informáticos podían ver las de todos. Dado que un usuario podría dar una prioridad superior a la que debería, se configuró el sistema de tal manera que un usuario no podía modificar sus propias notificaciones después de crearlas, pero un informático si podía. Además, esto impedía que un usuario añadiera o cambiase información en la notificación sin avisar al técnico asignado a su problema. Si un técnico quería ponerse en contacto con un usuario normal, podía hacerlo a través

de la herramienta de mensajería desarrollada en el anterior apartado.

Figura 4.18: *Lista de notificaciones de todos los usuarios*

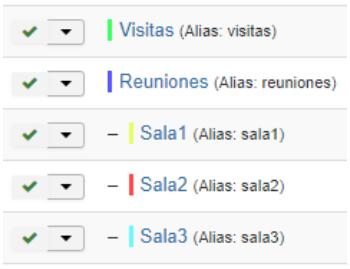
Una vez que se terminó de desarrollar este sistema para comunicar cualquier problema a los técnicos para que lo resolvieran, solo quedaba desarrollar el último requisito funcional.

4.6. Control de visitas al banco

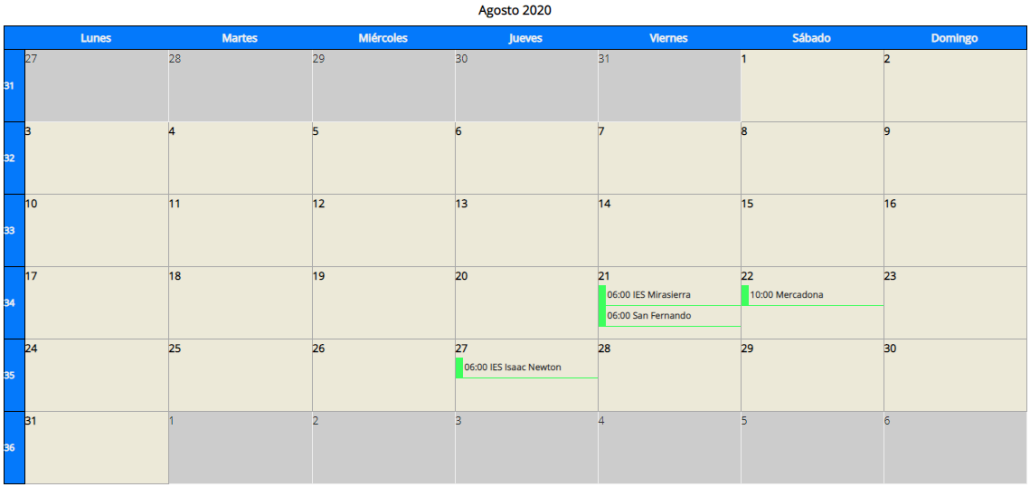
Con el resto de objetivos implementados, solo quedaba revisar el de menor valor para el cliente. Para implementar este objetivo, se debía diseñar un sistema sencillo que guardara las fechas de las visitas al centro y cuánta gente se esperaba que acudiera en cada una. Durante el desarrollo del proyecto ya se implementó un gestor similar que controlaba fechas, las reservas de las salas de reuniones, así que se utilizó la misma extensión para completar este objetivo, JEvents.

Dado que ya se utilizó este software anteriormente, la implementación del nuevo objetivo fue muy sencilla. Se creó un nuevo calendario en el que se guardarían las visitas, con su propia categoría para evitar cualquier otro tipo de evento. En este caso no se necesitaba hacer ninguna subcategoría. En los eventos no se podía generar un nuevo campo numérico para guardar la cantidad de visitantes esperada, pero tenían una descripción, así que se podía incluir dicha cantidad en ella. Esta opción era viable porque en los registros de la Fundación se detectó que no era habitual que se produjeran 2 visitas o más el mismo día y, en el caso de que pasara, el creador de la segunda visita encontraría la primera y podría comprobar cuantos visitantes iban a tener, evitando traer muchas personas al mismo tiempo.

El funcionamiento de este objetivo era el mismo que las reservas de salas, salvo que la categoría se asignaba automáticamente.



(a) Categorías finales del proyecto



(b) Calendario de visitas

Figura 4.19: *Gestión de visitas al Banco*

A pesar de que se avisó al Banco de que no había certeza de poder implementar este objetivo, se pudo hacer ya que había una extensión que resolvía el problema fácilmente. Con este objetivo se demuestra la efectividad de la extensión instalada y reafirma el hecho de que la aplicación podría incluir nuevas funcionalidades en el futuro fácilmente, cumpliendo uno de los requisitos técnicos más importantes del proyecto.

4.7. Máquina virtual y conexión a internet

Con la aplicación web terminada, había que preparar una máquina virtual que alojara la aplicación y escuchara las peticiones de los clientes. Para ello había que crear dicha máquina y configurarla para tener acceso a internet.

Al principio se optó por utilizar una máquina con un sistema Windows Server, ya que uno de los

servidores del Banco tenía instalado ese sistema, pero se acabó descartando porque era de pago. Se decidió instalar un sistema Linux porque no costaba nada instalarlo o utilizarlo, pero habían muchos sistemas con este kernel y no se sabía con seguridad cual era el más eficiente para el proyecto. Se evaluaron varios sistemas operativos y se compararon las ventajas e inconvenientes de cada uno, llegando a la decisión de utilizar Ubuntu Server.

Se encontró la página oficial de Ubuntu en la que se podía descargar el sistema, pero mientras se buscaba dicha página se halló otra que ofrecía imágenes de máquinas virtuales con sistemas Linux instalados, todas las máquinas que se ofrecían eran de código abierto. Esta página era Osboxes, y una de las máquinas que ofrecía tenía instalado Ubuntu Server, la versión 19.10. Está imagen estaba diseñada para importarse en un gestor de MV específico, VirtualBox. Debido a la experiencia previa con dicho gestor y que estaba la posibilidad de utilizar una máquina ya configurada para él, se decidió incorporarlo al proyecto con la máquina en cuestión.

Después de importarse en el gestor, había que configurar la máquina para tener acceso a internet. Mientras se buscaba una explicación para realizar dicha configuración, se encontraron muchas preguntas que trataban este asunto en varios foros, pero hubo una pregunta que hacía referencia a la misma máquina que se estaba utilizando en el proyecto. Se siguieron los pasos que se indicaban en ella, se configuró la máquina en el VirtualBox para utilizar un adaptador puente con la máquina física y se creó un fichero en una dirección concreta de la máquina virtual que indicaba la información necesaria para reconocer la red como un acceso a internet.

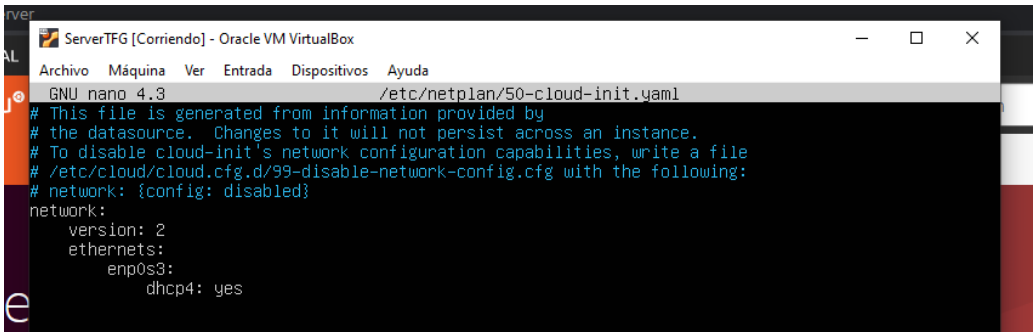


Figura 4.20: *Fichero de configuración de red de la máquina virtual*

Después de configurar la red y de confirmar el acceso a internet, se decidió instalar una interfaz gráfica para la máquina. Se tomó esta decisión porque los administradores del Banco que iban a gestionar la máquina no tenían mucha experiencia con un sistema Linux, por lo tanto una interfaz gráfica les permitiría manejarla con más facilidad.

Se instaló un paquete que gestionaba interfaces gráficas y permitía instalar varias en la máquina. El paquete era tasksel, y en este proyecto solo se utilizó para instalar un gestor de interfaces, SLiM, y una interfaz, GNOME. Después de instalar todos los paquetes y ficheros, se reinició la máquina y se confirmó que la interfaz gráfica se estaba ejecutando y que seguía teniendo acceso a la red.

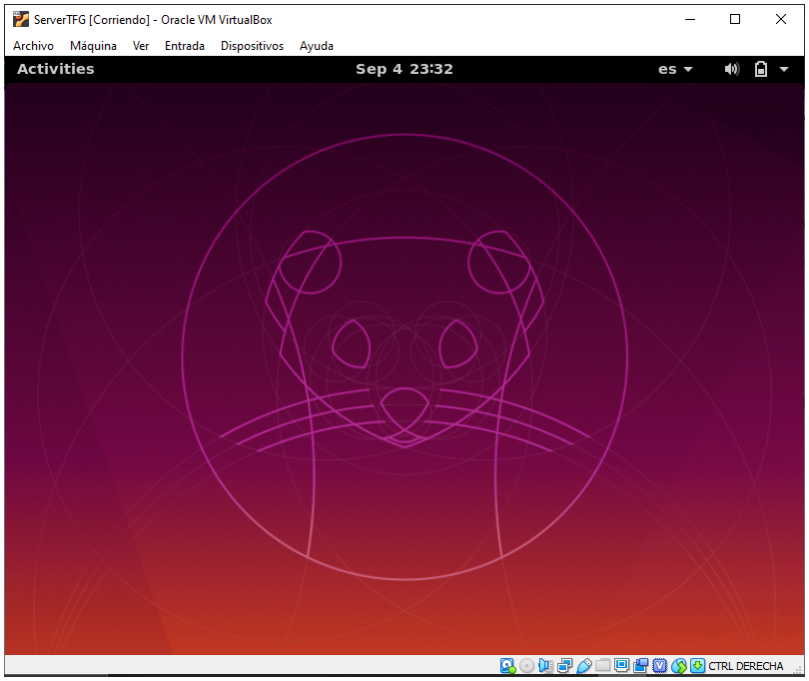


Figura 4.21: *Máquina virtual con la interfaz gráfica*

Después de confirmar el correcto funcionamiento de la máquina, solo quedaba instalar todo el software necesario en ella y trasladar la aplicación. Una vez que se hiciera esto, el sistema ya cumpliría todos los requisitos funcionales y técnicos del proyecto.

4.8. Instalar la aplicación web en la máquina virtual

En este punto del desarrollo ya estaba terminada la implementación de los objetivos del proyecto y la configuración inicial de la máquina virtual, lo único que faltaba era preparar la máquina con la aplicación y hacer las pruebas finales del proyecto.

Antes de trasladar la página, se revisó su presentación y se mejoraron algunos elementos que se habían ignorado, como algún campo de texto que estaba muy pegado a otro campo y detalles similares. También se revisó la máquina virtual y se cambió el fondo de escritorio para mantener la imagen de la empresa.

Con todas los cambios estéticos terminados, se buscó la forma de trasladar el software. Primero se instaló XAMPP en la máquina virtual para gestionar la información de la aplicación. Después se subió la aplicación web y la BD a una carpeta de Google Drive para acceder a ellas desde la máquina que las iba a alojar. Cuando se descargó y se importó la BD, se produjo un problema de

incompatibilidad en el sistema porque el fichero SQL era de una versión anterior a la instalada. Para solucionarlo se utilizó el comando "mysql-upgrade" que ofrece XAMPP en un script aparte para actualizar la BD al sistema de la máquina. Con ese comando ejecutado, se borró la BD en el sistema y se volvió a importar.

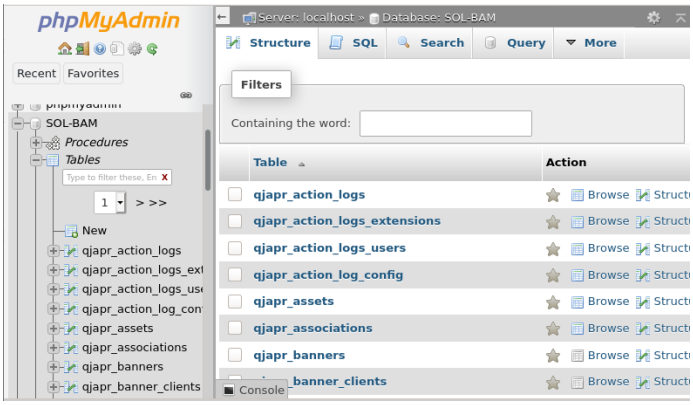


Figura 4.22: *Base de datos en la máquina virtual*

Por último , se importó la aplicación en la carpeta "htdocs" de XAMPP y se accedió al software desde el navegador para confirmar que la instalación se había realizado correctamente, sin embargo se produjeron unos problemas en el módulo de asistencia de voluntarios. Eran debido a que los scripts php generados creaban ficheros en el directorio "tmp" de la página cuando se ejecutaban y el sistema no permitía generarlos. Para solucionarlo se dieron permisos de escritura en la carpeta de la página web, el directorio "tmp" y el directorio "logs". Este último era necesario para ejecutar la aplicación en modo depuración y que se pudieran guardar los ficheros con el registro de la ejecución.

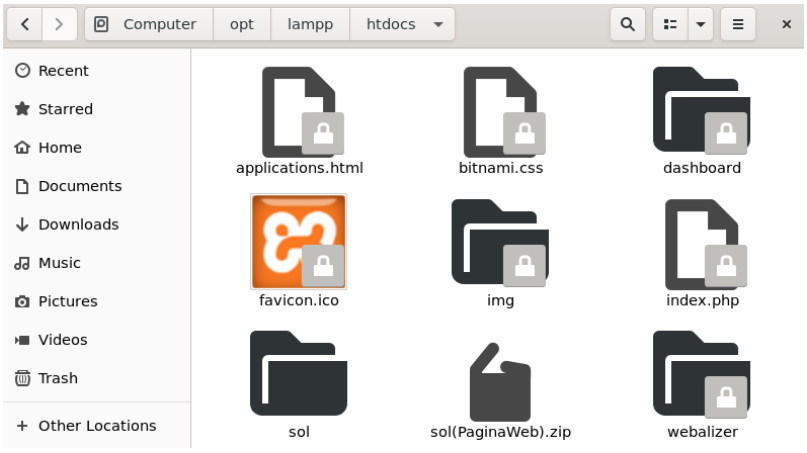


Figura 4.23: *Página web en el directorio "htdocs" de la máquina virtual*

Con todo el software trasladado a la máquina virtual y configurado para que se pudiera ejecutar desde el navegador, solo faltaba hacer las pruebas finales del sistema para confirmar el correcto funcionamiento de la aplicación.

4.9. Pruebas Finales

Dado que era una aplicación que se iba a utilizar desde muchos equipos con distintos sistemas y distintos navegadores, había que hacer unas pruebas de funcionamiento del software.

Estas pruebas consistieron en acceder al sistema y poder utilizar todas sus funcionalidades sin problemas. La máquina virtual estaba configurada para que cualquier equipo conectado a la misma red WiFi pudiera acceder al sistema, así que lo primero que se confirmó era que la máquina que estaba alojando la máquina virtual tenía acceso desde el navegador. Los navegadores que se utilizaron para las pruebas fueron Google Chrome, Microsoft Edge y Mozilla Firefox, que eran los navegadores más habituales de la Fundación. Se accedió con una cuenta de un usuario normal, con un informático y con un administrador y se utilizaron todos los módulos del sistema para confirmar su funcionamiento. También se accedió al backend con el administrador y con el super usuario.

Cuando se terminaron las pruebas en la máquina que alojaba el servidor, se hicieron pruebas en otros 3 dispositivos que también estaban conectados a la misma red. 2 de estos dispositivos tenían instalado un sistema Windows y el tercero tenía un Ubuntu. Las pruebas fueron las mismas, se accedieron con varias cuentas y se confirmó la funcionalidad de ellas. Para terminarlas, se simuló una conversación entre 2 usuarios de 2 equipos distintos y tampoco se produjeron problemas.

No se hizo ninguna prueba más compleja como una prueba de estrés porque no se sabía la arquitectura de los sistemas del Banco y cualquier prueba de este tipo de la página sería en un equipo diferente, por lo que no se podría asegurar que los resultados se repitieran cuando se trasladaran a los servidores del BAM.

Estas pruebas sirvieron para confirmar que la aplicación funcionaba correctamente, se podía acceder a ella desde distintos equipos y que podía satisfacer las necesidades del Banco de Alimentos de Madrid.

Capítulo 5

Conclusiones

Con el desarrollo del proyecto terminado y la aplicación funcional, se cumplió el objetivo principal del proyecto, diseñar un sistema que ayude a los empleados y voluntarios del BAM con su trabajo en el día a día. La aplicación se puede utilizar a la vez desde distintos equipos con navegadores diferentes y se puede modificar en el futuro para recoger necesidades que surjan con su uso.

En este trabajo se ha desarrollado un proyecto software que mantiene las pautas de proyectos que se realizan en empresas en el ámbito laboral, algunos de estos conceptos no se habían realizado en la carrera con este nivel de detalle y ha sido una experiencia nueva. Se tuvieron muchas reuniones con el cliente de la aplicación, se estudió el funcionamiento interno de su empresa y se extrajeron requisitos para marcar las pautas del proyecto y lo que se iba a abarcar, aparte de su posterior presentación al cliente, la justificación de porqué se hicieron las cosas cómo se hicieron y el feedback para conocer qué partes de la aplicación gustan más y qué partes se deben mejorar y cómo hacerlo.

También se ha aprendido que hay factores externos que pueden afectar al desarrollo de cualquier proyecto, es necesario identificarlos para preparar un plan de contingencia y que no afecten negativamente al desarrollo, pero pueden surgir eventos que no se valoraron. En estos casos hay que diseñar un plan en el momento para corregir el rumbo del proyecto. En el caso del SOL-BAM tuvo lugar la crisis del Covid-19 y la cuarentena, que afectaron al proceso de desarrollo y a la comunicación que se mantenía con el cliente. Cuando se produjo este incidente, se optó por detener temporalmente el desarrollo y continuarlo más tarde, aparte de modificar la fecha de entrega prevista a la convocatoria de septiembre.

A pesar de las complicaciones que surgieron, se pudo presentar la versión final del proyecto al BAM explicando qué necesidades se recogieron y qué funcionalidades incluía la aplicación. Cuando los responsables del Banco vieron dicha versión se mostraron complacidos con los resultados y redactaron una carta en la que se recoge su satisfacción con la aplicación.

Para concluir quiero recalcar el hecho de que este proyecto es un escenario real en el que el

sistema resuelve problemas reales y que el cliente acepta la propuesta mostrada, un hecho que me llena de orgullo ya que he podido utilizar los conceptos aprendidos en la carrera para diseñar algo que puede ayudar a mucha gente que necesite de los servicios prestados por el Banco.

5.1. Conclusions

With the development of the project finished and the application completely functional, the main objective of the project was achieved, design a system which could aid the employees and volunteers from BAM with their work. The application can be used at the same time from different computers with different web browsers and can be updated in the future so it can meet needs arising from its use.

In this thesis it has been developed a software project that maintains the guidelines of other projects carried by companies in their business, some of the concepts used haven't been applied during the degree with this detail and have been a new experience. Many meetings were held with the client, the internal process of the company was studied and the requisites of the project were extracted so that the guidelines of the project could be stablished and set said project in the correct way, apart from its subsequent presentation to the client, the justification of why everything was done the way they were and the feedback given so it could be known which parts from the application were appreciated and which other needed to be improved and how.

It has also been learned that there are external factors that can affect the development of any project, they need to be identified in order to prepare a contingency plan and stop them before they hinder development, however there may arise events that weren't considered. In this instances it's necessary to design a plan on the spot to correct the course of the project. In the case of SOL-BAM the Covid-19 crysis and the quarantine took place, which hindered the development of the project and the communication with the client. When this incident happened, it was decided to temporarily stop the development and continue it later, apart from modifying the delivery date to September.

Even with the obstacles that arose, the final version of the project could be presented to BAM explaining which needs were gathered and which functionalities were included in the application. When the Bank's representatives saw this version they were pleased with the results and drafted a letter expressing their satisfaction with the implementation.

To conclude I would like to emphasize the fact that this project is a real scenario in which the system solves real problems and the client accepts the proposal shown, a fact that fills me with pride since I have been able to use the concepts learned in the degree to design something that can help many people who need the services provided by the Bank.

5.2. Reuniones con el BAM y la cronología del proyecto

En todas las reuniones que se tuvieron con el Banco de Alimentos de Madrid estaba D. Juan Antonio Alonso Pomareda, Director de Informática, y gracias a su ayuda se pudo conocer el funcionamiento del Banco con mucho detalle y diseñar un sistema útil para la Fundación.

En la primera reunión, en abril del 2019, se enseñó una propuesta para optimizar el traspaso de la comida de donantes a los clientes que la necesitaban, como comedores sociales o iglesias. Esa idea fue descartada rápidamente porque ya tenían una aplicación para ello, sin embargo, el Director se mostró dispuesto a tener varias reuniones para explicar el proceso interno del Banco y detectar cualquier problema que tuvieran.

En este punto se buscaron varias metodologías para gestionar el trabajo de la forma más eficiente. Teniendo en cuenta que se necesitaba extraer todos los requisitos antes de seguir con el resto del proyecto, se decidió utilizar una metodología en cascada o waterfall. Después de esta decisión, empezaría la fase de extracción de los requisitos.

En las siguientes reuniones, el Director empezó explicando el proceso de demanda de comida. Tenían varios comedores sociales en la ciudad que se ponían en contacto con ellos para pedir comida. En muchos casos, los pedidos se repetían mensualmente, así que solo avisaban a la Fundación cuando querían modificar el pedido de un mes concreto. Con esto, el Banco sabía cuanta demanda tenía y podía organizar la comida que necesitaba. Cuando una persona individual se ponía en contacto con ellos para pedir comida, la redirigían a uno de los comedores, así en vez de repartir comida a individuos la repartían a locales, simplificando el proceso de entrega y ahorrando combustible de los camiones y tiempo.

En cuanto a las donaciones de comida, el Banco informaba a los donantes de puestos de recogida cercanos a su vivienda o de cualquier evento próximo en el que iban a recoger alimentos. La gran mayoría de estos eventos formaban parte de las Operaciones Kilo que organizaban periódicamente. Parte de las donaciones que recibían era dinero, por lo tanto, cuando no alcanzaban la cantidad de comida necesaria para satisfacer su demanda, podían comprar lo que necesitaban a empresas como Mercadona o Lidl.

También inventaron un sistema para recibir más dinero, “Apadrina una calle”, en la que una empresa pagaba una suma concreta para que nombraran una calle de los almacenes del Banco con su nombre y lo publicaran en su página web, algunas empresas que lo hicieron fueron Bankia, Carrefour y Google.

Una vez que se terminó con el proceso interno, en mayo, se empezó a revisar cómo controlaban cada parte del proceso en las oficinas. Se revisó qué herramientas utilizaban para la gestión de los almacenes, como se organizaban a los voluntarios y que problemas sucedían mientras hacían su actividad normal.

En octubre del 2019 se había terminado de extraer todos los requisitos y había confirmación del Banco de que estaban de acuerdo con la lista mostrada. A partir de este punto empezó la fase de diseño de la arquitectura.

Todo el proceso de selección de la arquitectura llevó mucho tiempo ya que se optaron por varias opciones. Se tenía claro que se necesitaba un sistema centralizado que procesara todas las peticiones de clientes, pero no estaba tan claro qué herramientas se iban a utilizar.

La primera opción fue diseñar 2 aplicaciones, una en el cliente programada a mano con algún lenguaje como Java o Python y otra en el servidor que escuchaba peticiones HTTP con Javascript o php, pero se descartó rápidamente porque llevaría demasiado tiempo hacer un sistema así, era muy probable que tuviera fallos graves de seguridad y los equipos del Banco abarcaban muchos sistemas operativos que se tenían que controlar.

Con el requisito de los equipos del Banco, se tenía que diseñar un sistema que procesara todo en el servidor y que el cliente solo enviara y recibiera información, así que se pensó en diseñar un servidor con Spring, un framework que permitía programar un servidor con Java y que tenía muchos protocolos desarrollados. Este software se iba a aprender con detalle durante el segundo cuatrimestre del curso, pero no se sabía nada de él en el momento, así que se tenía que investigar durante el primer cuatrimestre para estar seguro de que fuera de utilidad. También había otro problema, la aplicación sería de código cerrado y cualquiera que quisiera añadir una funcionalidad nueva tendría que acceder al código de la aplicación para modificarla, situación que no le interesaba al Banco.

La última opción fue utilizar un CMS, era un sistema mucho más fácil de manejar y de aprender, se utilizaba en muchas empresas para gestionar aplicaciones web y podía incluir nuevas funcionalidades fácilmente, luego era perfecto para el proyecto. Se optó por utilizar Joomla, se buscó que se necesitaba para instalarlo y algunos tutoriales para aprender a manejarlo.

En febrero del 2020 se presentó la arquitectura al tutor director del proyecto y, después de dar el visto bueno, se empezó con el desarrollo del proyecto. Durante este mes se intentó contactar con el Banco para informarles de la arquitectura de la solución, pero no estaban disponibles. No cogían el teléfono ni respondían a los correos. Un mes después empezó la crisis del Covid-19.

Durante esta crisis no se pudo trabajar con el proyecto debido a varios problemas personales, así que el desarrollo se detuvo temporalmente. En junio del 2020 se reanudó el trabajo y, después de insistir varias veces, se realizó una reunión con el Banco a finales de julio. En esa reunión se mostró un prototipo de la aplicación y se volvieron a confirmar las restricciones.

Se implementó el resto del proyecto durante el mes de agosto, se realizaron las pruebas finales del sistema y se enseñó una versión funcional con todos los objetivos cumplidos al Banco en una reunión telemática en septiembre. Los responsables que estuvieron en la llamada se mostraron muy satisfechos con lo que vieron y prepararon una carta para confirmar la veracidad del proyecto.

5.3. Alternativas a las tecnologías utilizadas

Durante el proyecto se evaluaron varias tecnologías que permitirían implementar la aplicación. Muchas de estas tecnologías se utilizaron al principio, pero se acabaron descartando porque daban conflictos con los componentes previos o se producía otro tipo de problemas. Sin embargo, había

tecnologías que se evaluaron y que no se utilizaron en ninguna parte del desarrollo, simplemente se descartaron por una opción mejor.

Respecto al desarrollo de la aplicación web, solo se encontró una opción válida para implementarla, un sistema CMS. En cuanto a qué sistema concreto, se optó por utilizar Joomla, pero habían más CMS que podían haberla desarrollado:

- **WordPress**: Es otra herramienta similar que permite generar y gestionar aplicaciones web. WordPress era el CMS más utilizado en el momento porque era más fácil de utilizar y disponía de más extensiones y plantillas, pero se escogió Joomla porque tenía una política más flexible con sus extensiones, lo que permitía configurar la página con mayor detalle y diseñar una solución más efectiva para el Banco.

- **Drupal**: Es otro CMS menos popular que las opciones anteriores. La diferencia de Drupal respecto al resto era su enorme potencial, pues estaba orientado a grandes aplicaciones sin necesidad de muchas extensiones externas. Se descartó porque era una herramienta muy potente y la curva de aprendizaje era más compleja, y no estaba pensada para una aplicación más pequeña.

Con el CMS se necesitaba un gestor de BBDD para controlar la información que estaba en el sistema. De las opciones evaluadas, cualquier gestor podía servir, pero se escogió XAMPP debido a que se había trabajado con él anteriormente. La otra opción barajada fue WampServer, un gestor similar que también podía controlar aplicaciones web.

Para la máquina virtual se revisaron varios sistemas operativos que estuvieran orientados a atender las peticiones de otros equipos. Todos los sistemas que se evaluaron eran Linux porque eran gratuitos. Si bien es cierto que se escogió Ubuntu Server por su facilidad de uso y su popularidad, lo que provocaba que hubieran más foros en internet que solucionaran problemas en el sistema, había más opciones:

- **Oracle Linux**: Es un servidor diseñado por Oracle. Se puede utilizar en muchos equipos hardware diferentes y es conocido por ser muy eficiente en la gestión de BBDD.

- **Fedora**: Es un sistema orientado a usuarios que no tienen mucha experiencia con Linux, pero, si la tuvieran, podrían configurarlo en función de lo que necesiten. Dispone de varias interfaces gráficas en función de lo que busque el usuario, como GNOME o KDE.

- **CentOS**: Es uno de los servidores más populares para gestionar aplicaciones. Se creó a partir de un sistema Red Hat Enterprise Linux y ofrece un entorno más estable que se ha ido actualizando desde hace muchos años.

Respecto al gestor de la máquina virtual, había un software que serviría para el proyecto, VMware. No se utilizó porque se había encontrado una imagen de una máquina virtual que estaba diseñada para VirtualBox.

Una vez explicadas las alternativas tecnológicas del proyecto en alto nivel, hay que comentar las extensiones y plantillas de Joomla que se habían barajado mientras se desarrollaba la aplicación web.

5.3.1. Extensiones y plantillas alternativas de Joomla

Joomla es un CMS muy popular y hay muchos usuarios que han desarrollado alguna extensión o plantilla, luego hay muchas opciones entre las que escoger para implementar lo que se quiera hacer. Sin embargo, hay veces en las que que no hay tantas posibilidades.

Durante el desarrollo de cada objetivo del proyecto, se ha utilizado la extensión más efectiva para implementarlo. En muchos de los objetivos, no había muchas extensiones válidas para implementar lo que se necesitaba o dieron problemas con componentes previos, por lo que no había muchas opciones entre las que escoger. Este no era el caso del segundo objetivo, el sistema para reservar salas de reuniones, ya que habían muchas extensiones que podían implementarlo.

Muchos de estos sistemas trataban ámbitos muy concretos que eran difíciles de extrapolar a lo que se buscaba, como era la gestión de un hotel o de una biblioteca, pero había otras que no eran tan difíciles de dirigir al ámbito del proyecto o que trataba un ámbito más genérico, como gestionar eventos de cualquier índole. Hay que comentar 2 extensiones que no se utilizaron y que podían implementar el objetivo en cuestión:

- **TableBooking**: Es una extensión que controla las mesas de un restaurante. El sistema organiza las mesas siguiendo un horario y se pueden hacer reservas dentro de los turnos disponibles. Si bien es cierto que era un género prácticamente idéntico al de las salas de reuniones, la extensión utilizada en el proyecto, JEvents, manejaba eventos generales, lo que permitía que la extensión simplificara futuras implementaciones del sistema.
- **EasyAppointments**: Es una extensión que permite hacer reservas en el sistema. Esta extensión se podía extrapolar a cualquier ámbito en el que se pudieran utilizar reservas simples. No se utilizó porque no era tan sencillo implementar reservas diferentes para cada sala y JEvents permitía hacer eventos generales si no se sabía en que sala se iba a realizar.

Una plantilla es un componente muy importante de una aplicación Joomla, ya que le da formato a la aplicación. Es lo que ven los usuarios cuando se conectan en el sistema para utilizarlo y es lo que indica cómo se van a estructurar el resto de elementos para mostrar la información que se quiera en cada página.

En el desarrollo del proyecto se buscó una plantilla que mantuviera una estructura parecida a la página web del BAM, y se escogió una que tenía una presentación muy similar, pero se encontraron más plantillas que podían servir en el caso de que la primera provocara algún problema:

- **as002076**: Es una plantilla diseñada por As Designing que tiene un formato muy parecido al del proyecto. No se escogió porque los tonos de la página no eran iguales y llevaría un trabajo extra cambiar la presentación.

- **JB Growth**: Una plantilla más simple diseñada por JoomlaBuff, tiene unos tonos similares a los utilizados en el proyecto. No se utilizó porque la plantilla estaba separada en paquetes para que se instale solo lo que se necesita y la instalación de varios paquetes era de uno en uno.

Buscar alternativas a las opciones escogidas permitió desarrollar el proyecto con más facilidad porque se utilizaban las mejores herramientas para la implementación de los objetivos y, si se producía algún problema grave que obligara a cambiarlas, había opciones útiles que se habían estudiado anteriormente.

5.4. Posibles implementaciones en el futuro

Esta aplicación está diseñada para poder incluir muchas funcionalidades de muchos tipos en el futuro. Durante las primeras reuniones con el Banco se hablaron de algunos escenarios que se podrían dar en el futuro y de algunas de las implementaciones que se podían realizar:

- Actualmente el Banco tiene unos dispositivos que pueden leer códigos de barras y algunos tipos de alimentos del almacén tienen uno, sin embargo no han terminado de establecer una estructura para etiquetarlos minuciosamente, o si iban a cambiar los códigos de barras por QR. Una posible implementación es, cuando se estableciera la estructura, desarrollar un sistema que permitiera a los teléfonos conectarse a la aplicación y leer los códigos de barras o QR para identificar las cajas con los alimentos.
- El sistema se desarrolló teniendo en cuenta solo las necesidades de la sede del colegio San Fernando. Se podrían recoger las necesidades del resto de sedes e incluirlas en el sistema y que cuando se conectara un voluntario solo pudiera ver lo relacionado con su sede. En esta implementación se deberían modificar parte de las funcionalidades desarrolladas en el proyecto.
- Actualmente solo hay usuarios normales, informáticos y administradores en el sistema y el uso de la aplicación es para ámbitos más generales que puede necesitar cualquiera, como informar de cuando está disponible alguien. Se podría estudiar con mayor detalle los departamentos del Banco para incorporar alguna necesidad suya en la aplicación que solo pudiera utilizar un nuevo tipo de usuario. Un ejemplo sería el departamento de donaciones y que tuvieran un sistema de recogidas que pudieran informar a los almacenes de la entrada de alimentos.

Estos puntos son una posible necesidad que se deba solventar en el futuro, pero en el presente del proyecto se necesitaba una herramienta que resolviera problemas que tenían en la gestión diaria del Banco. Estos problemas están recogidos en los objetivos del proyecto y la aplicación los cumple.

Capítulo 6

Bibliografía

[1] Ask Ubuntu. *unable to connect to internet with osboxes.org ubuntu server 19.10 image on virtualbox.* https://askubuntu.com/questions/1207323/unable-to-connect-to-internet-with-osboxes-org-ubuntu-server-19-10-image-on-virt

[2] AS Templates. *Free Joomla! Templates.* https://www.astemplates.com/free-joomla-templates-page1

[3] CMS2CMS. *User Roles in WordPress and Joomla Explained.* https://cms2cms.com/blog/user-roles-in-wordpress-and-joomla-explained/

[4] FrMoisesfr. *Curso básico de Joomla.* https://www.youtube.com/playlist?list=PLug2BNqcHKuK8oa6a1qHRGwou_fdhwuzj

[5] GNU. *GNU General Public License, version 2.* https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[6] Hosting Diario. *Microsoft Windows Server.* https://hostingdiario.com/windows-server/#Ventajas_de_Windows_Server

[7] Joomla. *Joomla Documentation^{TM}. https://docs.joomla.org*

[8] Joomla. *Joomla Documentation^{TM} : Developing an MVC Component/Developing aBasic Component. https://docs.joomla.org/J3.x : Developing_an_MVC_Component/Developing_a_Basic_Component*

[9] Joomla. *Joomla! Extensions Directory^{TM}. https://extensions.joomla.org/*

[10] Joomla. *GNU General Public License v2.* https://tm.joomla.org/gnu-general-public-license-v2.html

[11] Osboxes. *VirtualBox Images.* https://www.osboxes.org/virtualbox-images/

[12] PhoenixNAP. *How To Install A Desktop (GUI) On An Ubuntu Server.* https://phoenixnap.com/kb/how-to-install-a-gui-on-ubuntu

[13] ProgrammingKnowledge. *Cómo instalar XAMPP en Ubuntu 16.04 / Ubuntu 18.04 (Linux).* https://www.youtube.com/watch?reload=9&v=R5CUn5wGQGg&ab_channel=ProgrammingKnowledge

[14] Ubuntu Pit. *Best Linux Server Distro: Top 10 Compared and Our Recommendation.* https://www.ubuntupit.com/best-linux-server-distro-top-10-compared-recommendation/

[15] Webempresa. *¿Qué es un CMS? Conoce los mejores gestores de contenido.* https://www.webempresa.com/blog/que-es-cms-los-mejores-gestores-de-contenido.html

[16] WebsiteSetup. *Joomla Website Tutorial: Building a Website Using Joomla, Step-By-Step.* https://websitesetup.org/build-website-with-joomla/#blogpost

Apéndice A

Carta de satisfacción del BAM

Después de enseñarles a los responsables del Banco la aplicación completa, redactaron una carta en la que confirmaban que se habían recogido algunos de los problemas del Banco y que estaban satisfechos con el sistema presentado.

Se puede ver en detalle en la siguiente página.

D. Juan Antonio Alonso Pomareda, Director de Informática de la Fundación Banco de Alimentos de Madrid, acredita que D. Enrique Miguel Torrijos Gabriel, estudiante de Ingeniería de Software de la Universidad Complutense de Madrid, se reunió con nosotros en varias ocasiones para diseñar su trabajo de fin de grado que lleva por título S.O.L-BAM, dirigido a la solución que cubre ciertas necesidades de información de la gestión interna del Banco. Esta solución cubre nuestras expectativas.

Y para que conste, lo firma en Madrid a 1 de septiembre de 2020

Fdo.: Juan Antonio Alonso Pomareda
Director de Informática
Fundación Banco de Alimentos de Madrid.

Figura A.1: *Carta de aceptación del BAM*

Apéndice B

Manual de Usuario

Para utilizar la aplicación hay que descargar el fichero con la extensión ".ova" en un equipo que tenga un gestor de máquinas virtuales como VirtualBox o VMware y seguir los siguientes pasos:

1. Importar el fichero en el gestor de máquinas virtuales. No es necesario modificar la configuración de red para tener acceso a internet, pero si se quiere modificar es responsabilidad del usuario. Se importará la máquina virtual y se podrá acceder a la aplicación con ella.
2. Encender la máquina y acceder al sistema. El usuario de la máquina virtual es:

Nombre de Usuario	**Contraseña**
osboxes	osboxes.org

3. Cuando se ha encendido el equipo hay que abrir el terminal y ejecutar las siguientes instrucciones:

```
osboxes@osboxes:~$ sudo /opt/lampp/lampp start
[sudo] password for osboxes:
Starting XAMPP for Linux 7.4.9-0...
XAMPP: Starting Apache...ok.
XAMPP: Starting MySQL...ok.
osboxes@osboxes:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:0c:af:17 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.47/24 brd 192.168.1.255 scope global dynamic enp0s3
       valid_lft 41616sec preferred_lft 41616sec
    inet6 fe80::a00:27ff:fe0c:af17/64 scope link
       valid_lft forever preferred_lft forever
osboxes@osboxes:~$
```

Figura B.1: *Instrucciones en el terminal*

La primera instrucción enciende el servidor XAMPP, la segunda nos permite saber cual es la IP de la máquina, se puede ver en el "inet" de la red "enp0s3", en la captura anterior es "192.168.1.47".

4. Abrir el navegador en un equipo que esté conectado a la misma red WiFi que el dispositivo que esté alojando la máquina e introducir una de estas direcciones:

Frontend	**Backend**
http://IP-Máquina-Virtual/sol	http://IP-Máquina-Virtual/sol/administrator

En la parte "IP-Máquina-Virtual" se introduce la IP que apareció en el paso 3, en el ejemplo sería "192.168.1.47".

5. Necesitas introducir un usuario en el sistema para acceder, estos son los que hay por defecto:

Nombre de Usuario	**Contraseña**	**Grupo Principal**
Paco	paco	Registrado
Paco2	paco	Registrado
Paco3	paco3	Registrado
Rosa	rosa	Registrado
Pepe	pepe	Informático
Admin	admin	Administrador
Super user	superuser	Super Usuario

Solo el administrador y el Super usuario pueden acceder al backend.

Si se va a utilizar el sistema, se recomienda cambiar las contraseñas de los usuarios a otras más seguras.