UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Aplicación web para la gestión de un centro de ocio juvenil

MEMORIA PRESENTADA POR:

Ángela Blanes Martínez

GRADO DE INGENIERÍA INFORMÁTICA

Dirección: Pau Micó

Convocatoria de defensa: Septiembre, 2020

Contenido

Índice de tablas

Índice de ilustraciones

Resumen

Los centros de ocio juveniles han de ajustarse a normativas de regulación y uso cada vez más complejas. Además, en muchos de los casos, estos centros han de dar soporte a diferentes jerarquías de usuario, la cual cosa complica su gestión.

En este TFG se propone desarrollar una herramienta web que facilite la gestión de un centro de ocio juvenil en cuanto a usuarios, enfermedades asociadas, actividades y pagos realizados.

Resum

Els centres d'oci juvenils han d'ajustar-se a normatives de regulació i ús cada vegada més complexes. A més, en molts dels casos, aquests centres han de donar suport a diferents jerarquies d'usuari, la qual cosa complica la seua gestió.

En aquest TFG es proposa desenvolupar una eina web que facilite la gestió d'un centre d'oci juvenil quant a usuaris, malalties associades, activitats i pagaments realitzats.

Abstract

Youth leisure centers have to comply with increasingly complex regulations and regulations. In addition, in many cases, these centers have to support different user hierarchies, which complications their management.

In this TFG it is proposed to develop a web tool that facilitates the management of a youth leisure center in terms of users, associated diseases, activities and payments made.

1 Introducción

1.1 Antecedentes

El proyecto estaría dentro del área de desarrollo de aplicaciones web, y el principal motivo por el que se ha decidido realizarlo, es por una carencia existente a la hora de registrar y gestionar los usuarios de un centro juvenil, con sus respectivos datos de tesorería, actividades, enfermedades… ya que, hasta día de hoy, todo está plasmado en papel y muchas veces hay problemas de extravío de documentos, por no hablar de todo el tema de protección de datos. Además, de esta manera, se oferta otra vía de comunicación entre el centro y los usuarios, ya que podrán ver sus datos y mantenerse informados de las actividades que se hagan.

1.2 Objetivos

Este desarrollo se basará en:

- Permitir llevar un control informatizado del censo de personas que forman parte del centro juvenil: niños/as, padres/madres en el caso de ser menores de edad, monitores…
- Vincular enfermedades, alergias, intolerancias… de los usuarios y tenerlo centralizado y actualizado.
- Informar de actividades realizadas como acampadas, excursiones…
- Llevar los pagos del centro.
- Poder tener un registro de albergues, proveedores… y en caso de necesitarlo cualquier monitor, poder consultarlo.

1.3 Requerimientos

Como bien se comenta antes, en este caso, se ha contactado con un centro juvenil de Alcoy (Juniors Flor de Neu), y han expuesto las necesidades que ellos tienen a día de hoy, explicando su situación, los inconvenientes y las dificultades que los han llevado a tomar esta decisión.
Este centro, ahora mismo no tiene nada informatizado y muchas veces, se necesita cierta información de alergias, enfermedades de algún niño… que no se tiene en cualquier momento porque se almacena todo en papel en el centro donde están.
Además, al año se realizan una serie de pagos que ahora mismo están siendo gestionados con Excel y de esta forma, los tendrían actualizados y accesibles en cualquier momento.
Así que, mediante estas tomas de contactos, se han definido los siguientes requerimientos más detallados a continuación:

- Aplicación modular. Los diferentes módulos de gestión se podrán desarrollar independientemente y su integración en la aplicación no dependerá de su estado.

- Soporta tipos de usuario diferentes con permisos de acceso diferentes. Dependiendo del tipo, el usuario podrá acceder a diferentes módulos de gestión y tendrán permisos distintos de ver, editar o eliminar registros.

    - Usuario restringido a los datos propios y/o a los datos de interés general (niño, padre/madre, tutor)

    - Usuario con acceso parcial a datos del centro (monitor del centro)

    - Usuario con acceso total a todos los datos de la aplicación (jefatura del centro)

    - Usuario con acceso solamente a Secretaría (secretario)

    - Usuario general con control total (admin)

- Se desarrollarán los siguientes módulos de la aplicación:

    - Información general: consulta de información propia (enfermedades asociadas al usuario, relación parental o de hermanos, los datos de contacto...). Todos los usuarios verán sus propios datos menos los generales (secretaría, jefatura y admin)

    - Usuarios: Gestión de usuarios (altas/bajas/modificaciones, etc.). Solo los usuarios de jefatura, monitor y admin podrán tener acceso a este módulo.

    - Secretaría: Gestión económica (secretario del centro con acceso a los datos económicos de los usuarios). A este módulo solo accederá el usuario de tipo secretario, jefatura o admin)

    - Enfermedades: Gestión médica (enfermedades de los usuarios). Es importante que a este módulo tengan acceso los monitores, jefatura y admin.

    - Calendario: Gestión de actividades (calendario general de actividades, posibilidad de crear eventos con entidad propia, como campamentos). A este módulo tendrán acceso todos los monitores, jefatura y admin pudiendo crear, modificar o eliminar actividades. Los padres/madres/tutores y niños, solo podrán visualizarlo.

    - Proveedores: Gestión de proveedores (para listar los diferentes proveedores de material, albergues, eventos… con su dirección, teléfono, tipo…). Aquí podrán entrar los usuarios monitores, jefatura y admin.

2 Anteproyecto

2.1 Estado del arte

Como en el mercado actual, existe un gran número de aplicaciones con las que gestionar una base de datos mediante un interfaz gráfico, se ha estudiado e investigado qué aplicaciones podrían parecerse a esta idea y se han encontrado algunas similitudes con las siguientes, de las que hemos sacado alguna nueva idea que añadir al proyecto:

- **SuiteCRM**
  Es una aplicación de código abierto para servidores orientada a administrar clientes. Está programada en php y está construida su base a partir de la última versión de Sugar CRM. Además de contar con la última liberación de SugarCRM, en la que tendríamos los módulos base (cuentas, contactos, oportunidades, calendario, llamadas, reuniones, emails, tareas, documentos, campañas, empleados…) cuenta con unos cuantos módulos de más como puedan ser: citas, facturas, informes, eventos, mapas de Google, plugin de Outlook, encuestas… y unos cuantos más.

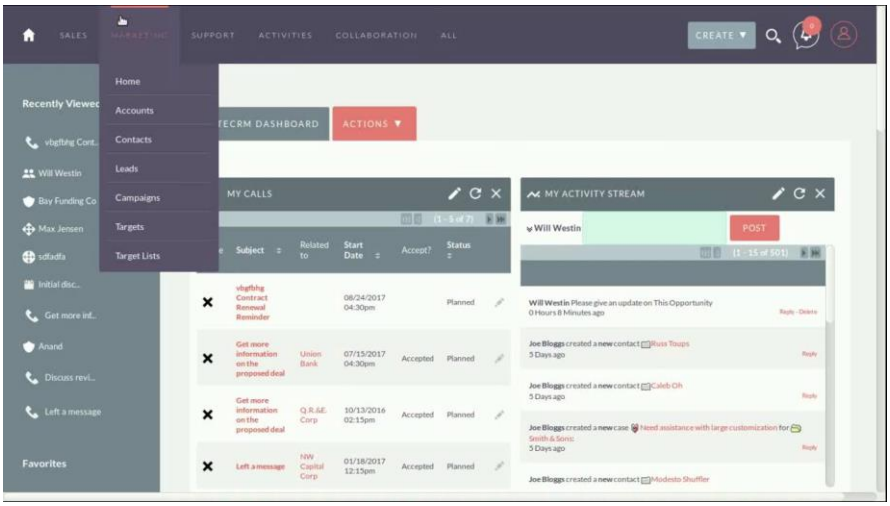

Ilustración 1. Entorno de SuiteCRM

- **CRM <u>Beyond</u> up**
  Es una herramienta CRM con la que se pueden hacer llamadas, enviar emails, organizar visitas y tener un control de los leads.
  Esta es una plataforma oncloud de gestión de clientes de alto rendimiento comercial y tiene un sistema de geolocalización con la que ver la ubicación de tus clientes para preparar las visitas. Se pueden automatizar procesos y es capaz de integrarse.
  Desde la app móvil guía tu actividad y registra cada acción, así la interacción con el cliente es compartida con otros departamentos y se pueden resolver enseguida las consultas que puedan surgir con una comunicación bidireccional.

Se pueden generar ofertas, crear contratos y cerrar ventas. Todos los documentos quedar guardados digitalmente y así no hará falta papel.

Ilustración 2. Entorno de CRM Beyond up

- **SugarCRM**
  Es un CRM que permite a los equipos de ventas, marketing y servicios poder colaborar durante el ciclo de vida de un cliente, para poder aportar las experiencias con valor, basadas en insights y predictivos, sin los inconvenientes de insertar manualmente datos. Sería muy parecida a SuiteCRM

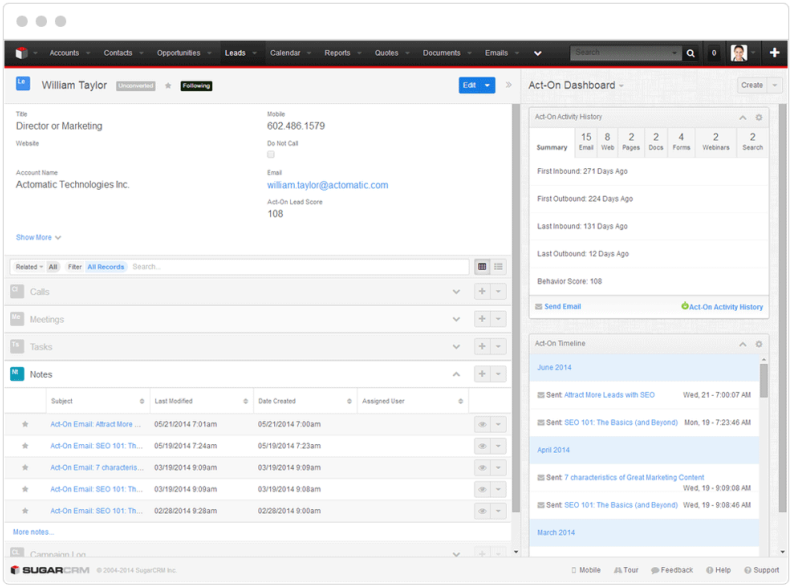

Ilustración 3. Entorno de SugarCRM

- **Zoho CRM**
  Es un CRM que habilita una red global con una gran cantidad de empresa en muchísimos países para más clientes potenciales, interactuar con ellos y aumentas los posibles ingresos que tenga cada uno. Además, tiene la posibilidad de realizar reuniones de ventas

y presentaciones, organizar seminarios web y luego convertir a los asistentes al seminario en clientes potenciales, con la posibilidad de obtener informes de los seminarios.

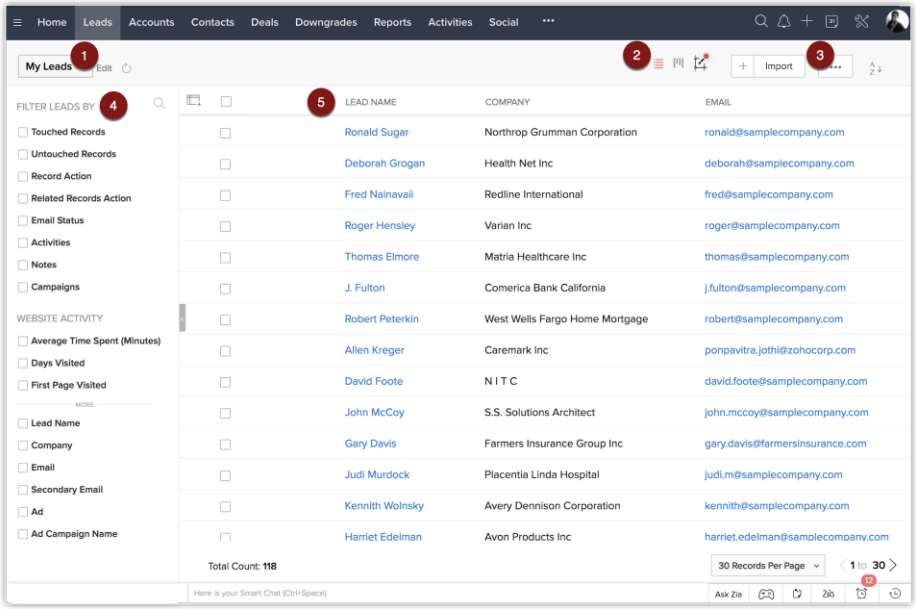

Ilustración 4. Entorno de Zoho CRM

2.1.1 Control de versiones

- **GIT**:
  Es un sistema de control de versiones distribuido, de código abierto, diseñado para gestionar los diversos cambios que se realizan sobre elementos de algún proyecto o configuración del mismo. Además, nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida.
  Yo, en mi proyecto, utilizaré un sistema de control de versiones online para evitar perder datos en el caso de que mi ordenador tuviera problemas y para poder revertir algún cambio que no funcione como era esperado o restaurar errores al desarrollar nuevas funcionalidades.

- **GITHUB**:
  GitHub es un servicio en el que se puede alojar código, publicarlo y compartirlo con otros desarrolladores para crear software. Además, tienes una interfaz gráfica basada en la web. Proporciona cierto control de qué usuarios acceden y funciones con la que colaborar.
  Su principal función es poder "bifurcar" un repositorio, es decir, poder copiarte un proyecto del que no tienes permiso de escritura, y bajo tu cuenta, poder editarlo. En el caso de querer compartirlo o actualizar dicho repositorio, se manda una petición al propietario y este será el encargado de poder fusionar los cambios.

- **Gitlab**
  Es un servicio web de control de versiones y desarrollo de software en el que colaboran los programadores basado en Git. Pero además de gestionar repositorios, se pueden alojar wikis y un sistema de seguimiento de errores con una licencia de código abierto.

2.1.2 Entornos de desarrollo

- **PhpStorm**
  Es un IDE para php, que proporciona un editor para php, html y JavaScript analizando el código sobre la marcha, para así poder prevenir los errores y con refactorizaciones automatizadas para código Php y JavaScript.
  Está escrito en Java y se puede ampliar en todo momento el IDE instalando complementos ya creados o generando unos propios.
  Se ha elegido este ya que en el trabajo lo estoy utilizando, y es muy cómodo para trabajar con el módulo de Git y poder trabajar directamente desde aquí con los dos.

- **Sublime Text**
  Es un editor de texto y de código escrito en C++ y Python y desarrollado como una extensión de Vim al principio.
  Se distribuye de forma gratuita pero no es de código abierto, pudiéndose obtener una licencia para usarla ilimitadamente, pero en el caso de no obtener dicha licencia, no causa ninguna limitación más que algún recordatorio o alerta.
  Soporta una gran cantidad de lenguajes como puedan ser C, C++, Java, JS, Python, PHP...

- **Visual Studio Code**
  Editor de código fuente desarrollado por Microsoft para Windows, Linux o Mac, incluyendo soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Además de poder personalizarse cambiando el tema del editor, los atajos del teclado, y las preferencias.
  Es gratuito, aunque la descarga oficial está bajo software privativo e incluye características personalizadas.

- **Atom**
  Es un editor de código fuente de código abierto para Mac, Linux y Windows con soporte para múltiples plugins escritos en Node.js y control de versiones Git integrado.
  Es una aplicación de escritorio que se ha construido utilizando tecnologías web. La mayoría de paquetes tienen licencia de software libre y están desarrollados por una comunidad de usuarios.

2.1.3 Lenguajes de Programación

- **Python**
  Es un lenguaje de programación desarrollado como proyecto de código abierto. Y como curiosidad, su nombre se debe a que el creador era fan de los Monty Python.
  Se trata de un lenguaje de programación en scripts, siendo un rival para Perl. Permite dividir el programa en módulos que se pueden reutilizar desde otros programas Python. También tiene una colección de módulos estándar. Y al ser un lenguaje interpretado, permite ahorrar proceso de compilado.

- **Java**
  Lenguaje de programación orientado a objetos que alcanzó su madurez cuando internet se hizo popular. La expansión de este lenguaje entre los desarrolladores y programadores ha sido vertiginosa.
  Hay muchas aplicaciones y sitios web que sin Java no funcionan y cada vez más. Es rápido, seguro y fiable. Su idea es poder realizar programas con la posibilidad de que se ejecuten en cualquier ambiente, siendo su portabilidad uno de sus logros.

- **PHP**
  Es un lenguaje de código abierto y uno de los más usuales a la hora de desarrollar web, ya que es posible incrustarlo en HTML. El código es ejecutado en el servidor, generando el HTML y enviándolo al cliente, además de que es posible configurar el servidor web para que todos los ficheros HTML se procesen con PHP.
  Es un lenguaje muy sencillo a la hora de empezar a programar, pero a su vez, tiene muchísimas características avanzadas para los desarrolladores que ya llevan un poco más y tienen algo de experiencia.
  Está centrado en la programación de scripts en el lado del servidor, y conlleva poder obtener datos de formularios, generar páginas dinámicas como en este caso…

- **HTML5**
  Es un lenguaje markup que se usa para estructurar y después presentar el contenido que queramos para una web. En este caso, es la quinta revisión del estándar, con lo que cuenta con la posibilidad de explotar usando menos recursos.
  Es un sistema dar formato a nuestras páginas y es la primera vez que HTML y XHTML se desarrollan en paralelo.
  HTML5 ofrece la posibilidad de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos.

- **JavaScript**
  Es un lenguaje de programación formal que da instrucciones a un ordenador para generar una serie de datos, en este caso, recursos dinámicos en una página web.
  Está basado en prototipos y orientado a objetos, utilizándolo normalmente en el lado del cliente, aunque también se utiliza en el lado del servidor.

Permite crear clases con las que luego, poder instanciarlas cuando sea necesario además de poder heredar características de otras clases, aunque cada objeto tenga sus propiedades y métodos.

- **jQuery**
  Es una librería o biblioteca basada en JS que facilita muchísimo a la hora de programar y se utiliza normalmente para añadir elementos interactivos en una página web sin tener que desarrollar mucha cosa.
  Una librería es una colección de recursos e implementaciones que están codificadas en un lenguaje en concreto y son funcionales, ya que al no tener que escribir tanto código, se ahorra tiempo...

- **CSS3**
  Se usa para definir la estética de un sitio web en un documento externo y con ello, está permitiendo que podamos, modificándolo, cambiar la estética del sitio web sin tocar nada el html que lo compone. Se puede especificar propiedades como el color, el tamaño de la letra, dar un tamaño concreto a un elemento… estableciendo una serie de reglas que indicarán cómo debe presentarse el documento.
  El W3C es el que se encarga de especificar que hojas de estilo servirán de estándar para navegadores o agentes de usuario.

- **Bootstrap 4**
  Bootstrap es un framework CSS de código abierto que ayuda a desarrollar webs más fácil y rápido ya que incluye plantillas con las que se puede modificar la tipografía, los botones, los formularios, menús… Además, permite crear interfaces de usuario compatibles con todo tipo de dispositivos, favoreciendo el design responsive, que mejora la experiencia de los usuarios y por ello, el posicionamiento.

- **Popper**
  Es una librería JS que sirve para poder añadir tooltips y popovers en html, ofreciendo muchas opciones para poder personalizarlo. Es modular y contiene diferentes plugins para características.

2.1.4 Servidor de bases de datos

- **MySQL Server**
  Se trata de un sistema de código abierto para la gestión de bases de datos relacionales respaldado por Oracle. Además, se puede ejecutar en la mayoría de los sistemas operativos…
  MySQL suele estar asociado la gran mayoría de las veces con aplicaciones web, y se basa en un modelo cliente-servidor donde el núcleo es el servidor MySQL, que gestiona los comandos de la base de datos.

Permite almacenar datos y que se pueda acceder a ellos por medio de diferentes motores de almacenamiento y poder replicar tablas de datos y particiones para un mejor rendimiento.
En este caso, he utilizado MySQL Workbench para gestionar las tablas de mi bbdd.

2.1.5 Servidor web

- **Nginx**
  Es un servidor web que llega con unos requerimientos técnicos un poco más complejos que su competidor Apache, por lo que es un poco más veloz y en el mundo del hosting es algo que se mira con muy buenos ojos.
  La mayor desventaja que tiene es la compatibilidad para las reglas de Apache que usan otros sitios, haciendo que a la hora de pasar de uno a otro no sea tan sencillo, teniendo que traducirlas al lenguaje propio de Nginx.

- **Apache**
  Este es un servidor web http de código abierto gratuito, con el que cuando se quiere visitar una web, se ingresa el nombre de dominio en la barra de direcciones y este envía los archivos solicitados. Además de ser multiplataforma y poder trabajar en varios sistemas operativos con un buen rendimiento. Soporta SSL y TLS, y puede dar soporte a lenguajes como Php, Python...
  La función de un servidor web es que se aloja en un ordenador con conexión a internet a la espera de que se le haga alguna petición y poder responder (por ejemplo, acceder a una web y enviar el código html por transferencia de datos en red.

2.2 Estudio de propuestas

En todas las propuestas que se especifican a continuación, además se ha de tener en cuenta en qué servidor se va a instalar, ya que contamos con dos posibles opciones valoradas por la posibilidad que el centro juvenil tiene ahora mismo:

- Servidor propio en el que poder alojar el sitio web, ya sea el de la propuesta 1 como el de la 2, el cual deberíamos coger uno de los equipos de los que ya disponemos o comprar uno para hacerlo servir de servidor.

- O contratar un hosting en el que alojar el sitio web y tener que pagar dicho hosting, con el mantenimiento que ello conlleva.

2.2.1 Propuesta 1

La primera propuesta es desarrollar la aplicación web dinámica, que permita la interacción del cliente con ella.
Una aplicación web dinámica requiere de una lógica de programación, ya que es un poco más compleja que una aplicación web estática desde el punto de vista técnico porque el contenido y la información de esta, va a estar actualizándose cada vez que se inicie, además de que se van a ir agregando registros periódicamente.
Los lenguajes más comunes utilizados para una aplicación web suelen ser Php y JavaScript, ya que, además, con ellos también se puede modificar el diseño de la aplicación.

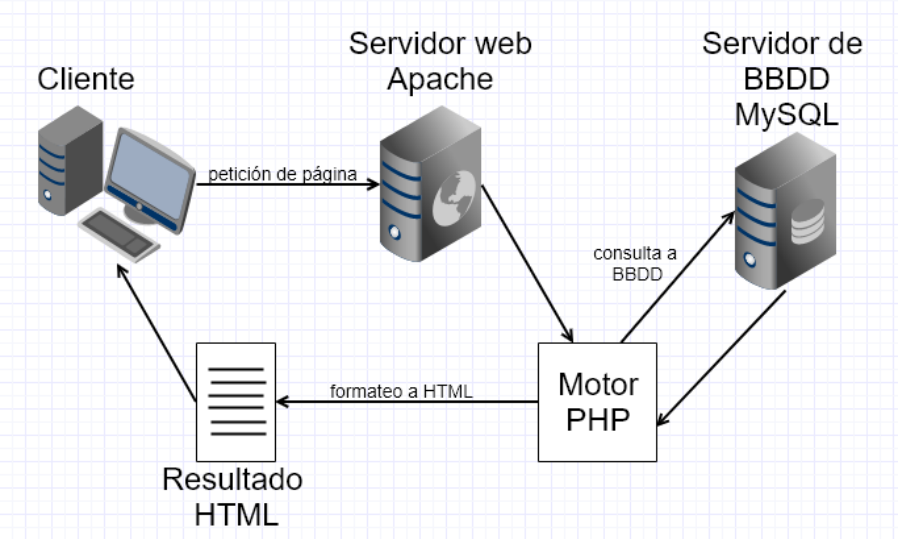

Ilustración 5. Esquema de la arquitectura de una aplicación web dinámica

- Control de versiones con GIT
- IDEs: PhpStorm
- Lenguajes de programación: CSS 3, HTML5, PHP, JavaScript y Bootstrap
- Servidor de bases de datos: MySQL Server
- Servidor web: Apache

2.2.2 Propuesta 2

La propuesta 2 sería realizar la aplicación web contando con un gestor de contenido como podría ser Joomla, WordPress...
Aplicaciones que son más comunes cuando el contenido debe actualizarse constantemente o se han de ir añadiendo contenido adicional. Y al tener un gestor (CMS), se pueden añadir, modificar o actualizar los contenidos.
Las páginas de periódicos o blogs serían ejemplos con los que sería perfecto utilizar un gestor de contenido digital.

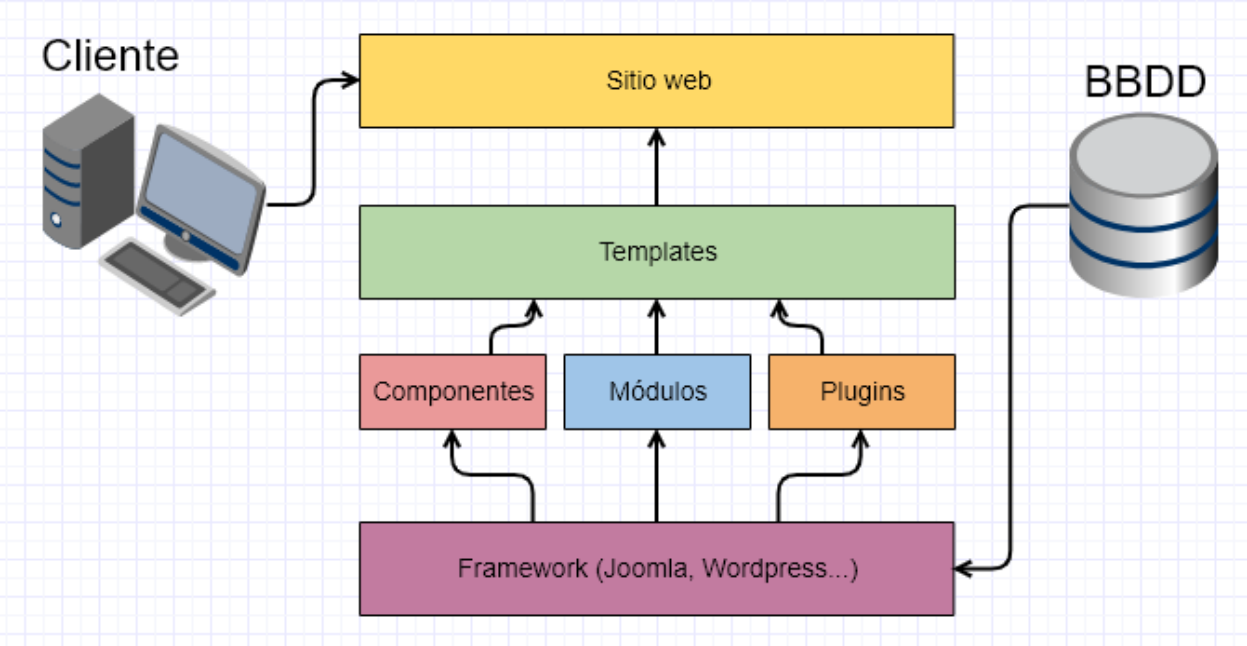

Ilustración 6. Esquema de la arquitectura de una aplicación web con gestor de contenido

Ilustración 7. Funcionamiento de un CMS

2.3 Justificación

Finalmente se ha escogido la primera de las dos opciones por varios motivos, aunque las dos son completamente válidas.
Además de por las ventajas que tiene hacerla sin un gestor de contenido, en el trabajo estoy desarrollando una aplicación web para la gestión interna del contenido de la web muy parecida y me ha servido para adquirir conocimientos que luego podré aplicar a la hora de implementarla y mejorarla. Es una inversión de adquisición de conocimientos que me vas a ayudar en un futuro próximo a la hora de implementar algunas de las funcionalidades que se necesitan.

Respecto a las cosas positivas que tienen las aplicaciones web dinámicas, podrían ser:

- Que la flexibilidad a la hora de desarrollarla es total, ya que se puede hacer cualquier cosa que se nos ocurra.

- Es más exclusiva la aplicación ya que no está sobre la misma base que muchas otras y le da esa originalidad que a lo mejor con un CMS no puedes tener.

- Y el precio del alojamiento puede ser muy bajo, pero con la responsabilidad de que el desarrollador ha de hacer una aplicación web optimizada.

Cierto es que, con un gestor de contenidos, en el caso de esta aplicación en la que íbamos a gastar Wordpress, también existen pros como, por ejemplo:

- Que tiene una base muy sólida y va mejorando constantemente.

- Es modular también y el mantenimiento, desarrollo y administración son de bajo coste.

Pero en este caso, como es una aplicación interna del centro juvenil y queríamos una implementación más personalizada, además de que no es necesario actualizar el contenido constantemente, me he decantado por hacerla dinámica sin contar con un gestor.

Además, también se ha decidido que como con los equipos que contamos, no tienen las características para poderlos hacer servir de servidor, y suele irse la luz en los locales en los que lo instalaremos, se ha decidido contratar un hosting para que la aplicación no de problemas y siempre esté activa por si se necesitase en cualquier momento, y dejar de depender de las condiciones que haya con los equipos de los locales.

2.3.1 Estimación de recursos

Voy a dividir la estimación en dos, ya que voy a diferenciar lo que se va a utilizar durante el desarrollo y lo que nos hace falta al ponerlo en producción.
Para el desarrollo, se utilizará:

- En la parte software:

    - El XAMPP, que es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl

    - PhpStorm, con el que contamos con la licencia en mi caso, y será el IDE con el que desarrollaré la aplicación.

    - MySQL Workbench, para cualquier consulta a BBDD ya que me parece más cómodo que estar entrando a phpmyadmin en el navegador y todo es mucho más intuitivo.

- En la parte hardware:

    - Como voy a utilizar mi ordenador para ello, será un portátil con un procesador Intel Core i7, con 16 GB de RAM y una memoria de 1 TB en disco y el sistema instalado en una SSD de 250 GB. Aunque es importante recalcar que con menos RAM y un procesador más inferior, también funcionará.

- Y, además, aquí valoro también algunos aspectos más como:

    - El internet utilizado, que, en mi caso, he tenido que contratar el internet por Radio/Wimax con una velocidad de 30 Mb, porque no tenía internet hasta el momento.

    - Las horas que se invertirán en este proyecto, en las que teniendo en cuenta que se trabaja, he podido hacer un cálculo por encima de más o menos unos dos meses y medio.

    - El gasto de luz, que habría que ver las horas y el consumo del PC para sacar la parte proporcional.

    - Y el gasto y desgaste del equipo.

Y en la puesta en marcha en producción, necesitamos solamente:

- Principalmente un ordenador con acceso a internet, que vuelvo a utilizar el mío con el mismo internet que en la parte de desarrollo.

- Un Hosting en el que instalar la aplicación y poder acceder a ella, en el que normalmente, viene también la posibilidad de alojar la base de datos en el mismo sitio.

- Y un certificado SSL para que la aplicación esté cifrada ya que estaríamos trabajando y manejando datos de usuarios como DNI, correos...

2.3.2 Impacto económico

Bien, pues a continuación, entramos en detalles económicos ya que, con lo estimado anteriormente, he calculado el coste de todos los recursos.

Para el desarrollo, se ha estimado el tiempo de 2 meses y medio de trabajo, a 6 horas al día, de lunes a viernes, y nos salen los siguientes datos:

Tabla 1. Estudio económico del desarrollo de la aplicación

CONCEPTO	**Precio/ mes**	**Precio total**
Internet Tarifa 30 MB → 23,80€/mes 40% de uso	9,52 €	23,8 €
Horas de trabajo 20€/hora 6 horas/día → 5 días/semana (55 días)	2.400€	6.000€
Gasto de luz 6 horas/día Tarifa plana → 63,59€/kW año → 0,14€/kWh Consumo PC → 1,65kW/día (6 horas → 0,6kW)	1,68€	4,2€
PhpStorm Licencia mensual 19,90€ ← Licencia anual 199,00€	19,90€	49,75 €
XAMPP (licencia gratuita)	-	-
MySQL Workbench (licencia gratuita)	-	-
Hardware PC Intel Core i7, 16GB de RAM, 1TB en disco, 250GB SSD	-	-
TOTAL	2.431,1 €	**6.077,75 €**

Para la puesta en producción, sí que deberemos investigar en el mundo de la compra de un hosting, y encontramos varias alternativas entre los más conocidos:

Tabla 2. Tabla comparativa de los precios de diferentes hostings

HOSTING A CONTRATAR	Precio/mes	Precio total
Hostalia Dominio Certificado SSL Hasta 800 MB de RAM 100 GB de disco SSD 25 usuarios FTP 25 bases de datos MySQL y PostgreSQL Tarifa anual → 126,90€/año	9,99€	19,98€
Hostinger ← Dominio con sitios web ilimitados Certificado SSL Cuentas FTP ilimitadas Bases de datos MySQL ilimitadas Integración con GitHub Copias de seguridad semanales Tarifa Premium → 45€/año	2,15€ (48 meses) **3,75€** (12 meses)	**45,00€**

Por lo que, ante las necesidades y los precios, he decidido quedarme con Hostinger, y seguir adelante con la instalación, ya que además incluye el certificado SSL, que también debía instalarse.

Además, para la realización de este proyecto, a continuación, se definen las fases y el tiempo estimado de dedicación a cada una de ellas, superponiéndose algunas ya que son fases que han de ir a la par.

.

Nombre de tarea	Duración	Comienzo	Fin
Planificación			
Plan de trabajo	4 días	lun 18/05/20	jue 21/05/20
Investigación			
Investigación y estudio de requerimientos	6 días	vie 22/05/20	vie 29/05/20
Diseño			
Analisis y diseño	13 días	sáb 30/05/20	mar 16/06/20
Implementación			
Codificación	20 días	mié 17/06/20	mar 14/07/20
Pruebas			
Testing	9 días	jue 02/07/20	mar 14/07/20
Documentación			
Memoria	13 días	mié 08/07/20	vie 24/07/20

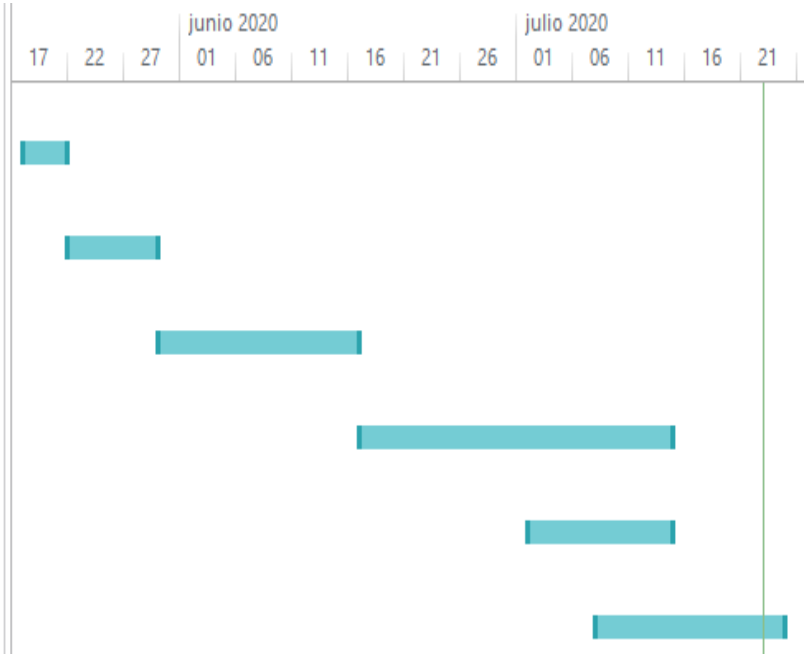

Ilustración 8. Estimación de las diferentes fases del proyecto

2.3.3 Propuesta final

En base a la arquitectura anteriormente detallada, a la estimación de recursos y al impacto económico que he podido ver, la solución que más se ajusta a las condiciones del centro, en relación a los recursos disponibles, la posibilidad de nuevas adquisiciones y el estado económico de esta (porque es un centro juvenil de voluntariado), sería contratar el hosting con 'Hostinger' ya que nos incluye el certificado SSL también, y nos ahorraríamos 12€ más o menos que es el coste del certificado por separado. En 'Hostinger' podríamos también administrar la base de datos y tenerlo todo junto.
En cuanto a la parte de desarrollo, PhpStorm es una buena decisión al contar ya con la licencia y poder utilizarla, y el resto, es todo software libre y por lo tanto gratuito.
Y, por último, con el internet que he contratado sería suficiente, aunque si se tienen más Mb pues mejor, pero no es necesario.

3 Implementación

3.1 Entorno de desarrollo

Para el desarrollo de esta aplicación, se tuvieron en cuenta varios factores a la hora de elegir el entorno de desarrollo, pero finalmente me decanté por PhpStorm por los siguientes motivos:

- El primero de todos es porque es el que estábamos utilizando en el trabajo y pensé que era una manera de aprender su funcionamiento ya que yo era la primera vez que trabajaba con él. Pensé que era una buena forma de entenderlo si además de en el trabajo, lo utilizaba para realizar esta aplicación web.

- El segundo motivo por el que me decidí por él fue porque tenía intención de trabajar y llevar un control de versiones con Git, y vi que PhpStorm trabajaba con GitHub y desde el mismo IDE he podido estar haciendo los commits, las subidas y las actualizaciones sin tener que estar abriendo otros programas. Además de que no hay muchos plugins que instalar porque todas las funcionalidades necesarias vienen por defecto.

- El tercer motivo fue relacionado con el trabajo también ya que contaba con la licencia del IDE por parte de la empresa y pensé en aprovecharla.

- Y, por último, contaba con una serie de características que me facilitaron muchísimo a la hora de programar como:

    - La asistencia de código inteligente, con la que se verifica que el código estaba bien.

    - Navegación de código inteligente, donde el IDE detectaba a que sitio del código hacía referencia y me llevaba allí.

    - Y se podía refactorizar el código para renombrar, mover, eliminar…. muy rápidamente.

Para montar el entorno local, se ha utilizado XAMPP ya que tendría el servidor apache, un servidor MySQL e intérpretes de lenguajes como php, que es el que he utilizado.

3.2 Implementación práctica

En mi caso, se trata de una aplicación web con una arquitectura de 3 capas.
Normalmente, las aplicaciones web muestran un esquema a tres niveles, donde la primera capa incluiría el navegador y el servidor web, encargado de darle a los datos el formato que toca. La segunda capa sería la lógica de negocio, es decir, scripts y funcionalidades varias… Y, por último,

la tercera capa sería la encargada de darnos los datos que necesitemos para poder ejecutar la aplicación.

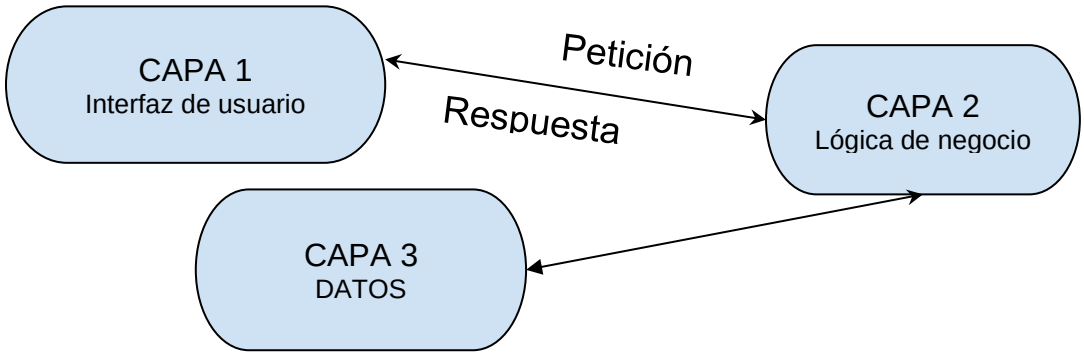

Ilustración 9. Esquema de la arquitectura de 3 capas

Se ha escogido esta arquitectura porque al estar dividido en tres capas, se puede modificar perfectamente cualquiera de las tres sin afectar a las otras. Además, las llamadas que hacemos de la interfaz a la capa 2, son más flexibles porque solo se necesita pasar los datos a esa capa.
Por lo tanto, a continuación, se especifica con más detalle las diferentes capas que se han diseñado.

3.2.1 Diseño del producto

En este caso, estaríamos hablando de la capa 3, es decir, el acceso a datos.
El esquema de la base de datos de esta aplicación web estará compuesta por 17 tablas.
A continuación, se adjunta el modelo entidad-relación y más abajo, una imagen de la bbdd.

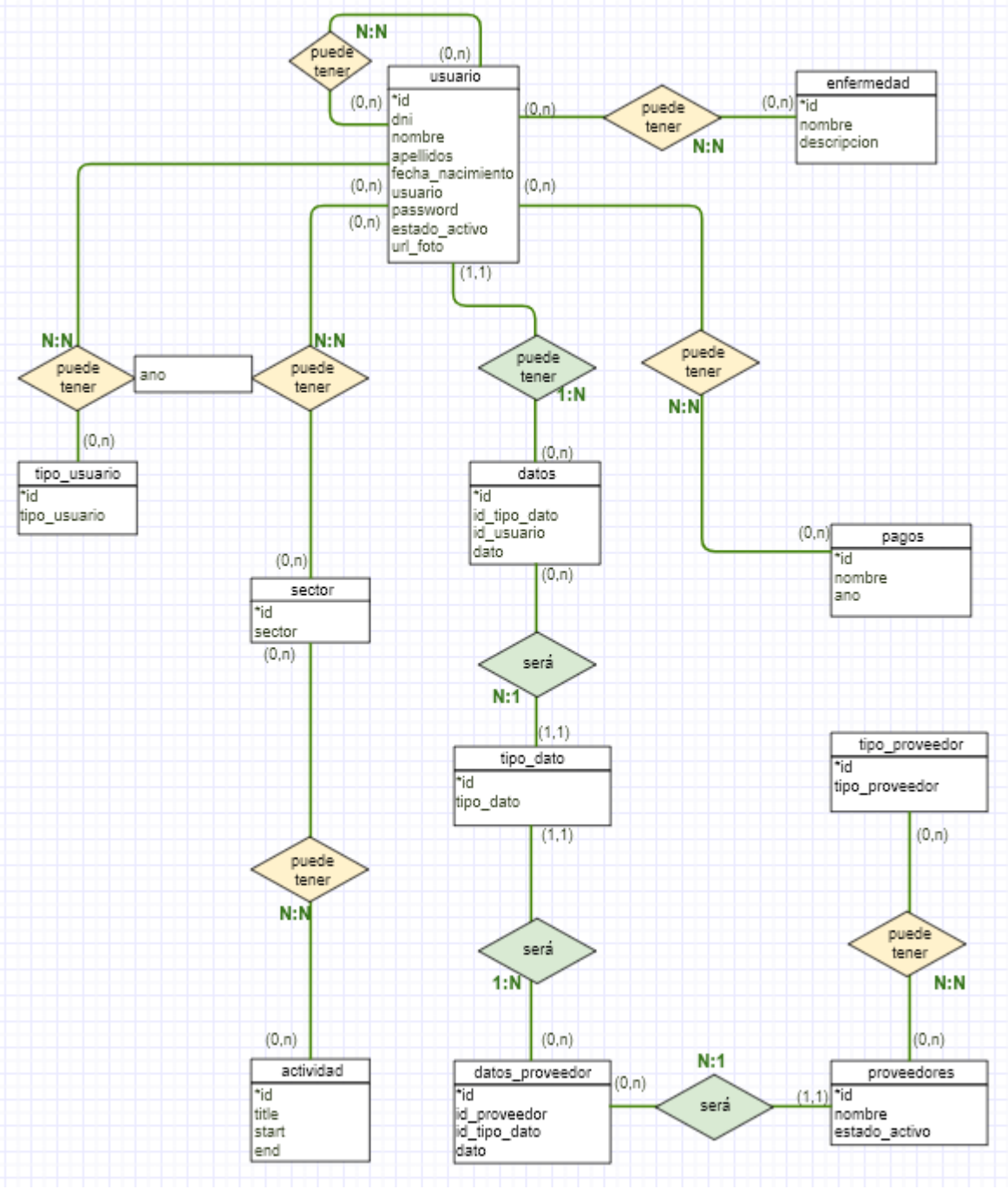

Ilustración 10. Modelo entidad-relación de la base de datos

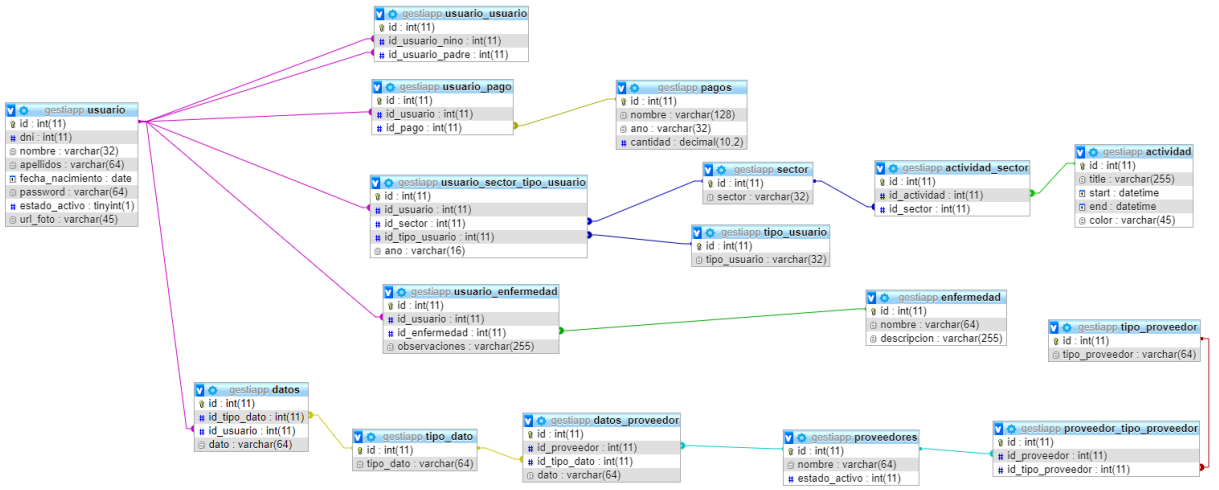

Ilustración 11.Esquema de la base de datos

3.2.2 Maquetas

A continuación, estaríamos en la capa 1, la interfaz, y defino una idea principal de cómo deberán ser las ventanas de la aplicación.
La aplicación web tendrá una estructura general en la que podemos ver un menú de navegación a la izquierda en el que se mostrarán arriba de todo el logo de la aplicación y luego todos los módulos existentes y a los que el usuario tenga acceso; y la parte central será el contenido, que es lo que irá cambiando dependiendo de en qué sección del menú nos encontremos.

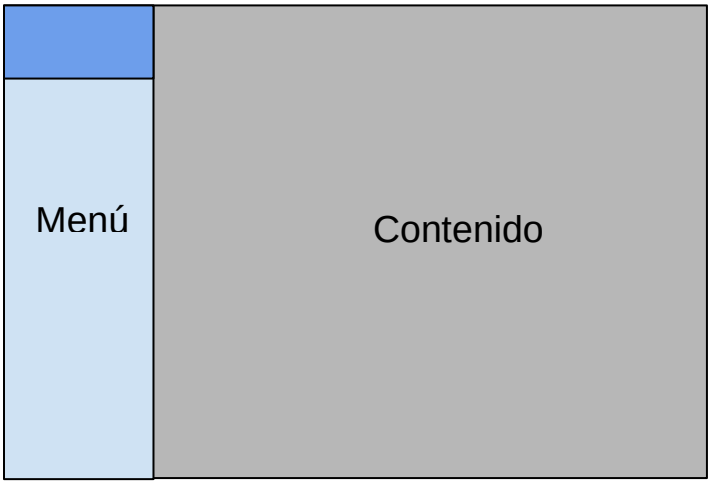

Ilustración 12. Diseño de la interfaz de la aplicación

Primero, adjunto el diseño del inicio de sesión, en el que, con un usuario y una contraseña proporcionados, se podrá acceder a la aplicación.

Ilustración 13. Diseño del Login

En este Login, se comprobará primero si hay una sesión iniciada y si es así la cargará; si no existe sesión, comprobará si el usuario existe (es decir, si existe el DNI del usuario y contraseña) y si el estado del usuario está activado.
Enseguida, se logueará y como se puede ver a continuación, se guarda toda la información del usuario que se ha logueado en sesión.

```
//Comprobar de envío el formulario
if (isset($_REQUEST['login'])) {
   $usuario = mysqli_real_escape_string($con, $_REQUEST['usuario']);
   $password = mysqli_real_escape_string($con, $_REQUEST['password']);
//Sacar el curso actual en el que se está
   $mes_actual = date("n");
   $ano_actual = date("Y");
   $ano_anterior = $ano_actual-1;
   $ano_proximo = $ano_actual+1;
   if ($mes_actual < 9){
       $curso_actual = $ano_anterior.'/'.$ano_actual;
   } else {
       $curso_actual = $ano_actual.'/'.$ano_proximo;
   }
   $_SESSION['ano_actual'] = $curso_actual;
//-------------------------------------------------USUARIO-------------------------------------
-------------------------
   $datos_usuario = mysqli_query($con, "SELECT * FROM usuario WHERE dni = '".$usuario."' and
password = '".$password."';");
   //Comprobación inicio de sesión
   if ($row = mysqli_fetch_array($datos_usuario)) {
       if($row['estado_activo']==1){
           $_SESSION['usr_id'] = $row['id'];
           $_SESSION['usr_nombre'] = $row['nombre'];
           $_SESSION['usr_apellidos'] = $row['apellidos'];
           $_SESSION['usr_dni'] = $row['dni'];
           $_SESSION['usr_fecha'] = $row['fecha_nacimiento'];
```

```
            $_SESSION['usr_url_foto'] = $row['url_foto'];

//-------------------------------------------------SECTOR y TIPO USUARIO-----------------------
--------------------------
            $sector_usuario = mysqli_query($con, "SELECT S.id, S.sector from sector S inner join
usuario_sector_tipo_usuario USTU on USTU.id_sector = S.id where
USTU.id_usuario='".$_SESSION['usr_id']."' and ano = '".$_SESSION['ano_actual']."';");
            if ($row1 = mysqli_fetch_array($sector_usuario)) {
                $_SESSION['usr_sector'] = $row1['sector'];
            } else {
                $_SESSION['usr_sector'] = '';
            }
            $tipo_usuario = mysqli_query($con, "SELECT TP.id, TP.tipo_usuario from tipo_usuario
TP inner join usuario_sector_tipo_usuario USTU on USTU.id_tipo_usuario = TP.id where
USTU.id_usuario='".$_SESSION['usr_id']."' and ano = '".$_SESSION['ano_actual']."';");
            if ($row2 = mysqli_fetch_array($tipo_usuario)) {
                $_SESSION['usr_id_tipo'] = $row2['id'];
                $_SESSION['usr_tipo'] = $row2['tipo_usuario'];
            } else {
                $_SESSION['usr_id_tipo'] = '';
                $_SESSION['usr_tipo'] = '';
            }
//-------------------------------------------------ENFERMEDADES-------------------------------
--------------------------
            if ($_SESSION['usr_id_tipo'] == '1' || $_SESSION['usr_id_tipo'] == '2') {
                $enfermedades = mysqli_query($con, "SELECT nombre from (SELECT U.id, E.nombre
FROM usuario U INNER JOIN usuario_enfermedad UE ON UE.id_usuario = U.id INNER JOIN enfermedad E
ON E.id = UE.id_enfermedad) as E where E.id = '" . $_SESSION['usr_id'] . "'");
                $enfermedadesArray = array();
                while ($enfermedad = mysqli_fetch_array($enfermedades)) {
                    array_push($enfermedadesArray, $enfermedad['nombre']);
                }
                $_SESSION['usr_enfermedades'] = $enfermedadesArray;
            }
//-------------------------------------------------DATOS---------------------------------------
--------------------------
            $datos = mysqli_query($con, "SELECT tipo_dato, dato from(SELECT D.id_usuario
,TD.tipo_dato, D.dato FROM datos D INNER JOIN tipo_dato TD ON TD.id = D.id_tipo_dato) as X
where X.id_usuario = '".$_SESSION['usr_id']."'");
            $datosArray = array();
            while ($dato = mysqli_fetch_array($datos)){
                array_push($datosArray, $dato);
            }
            $_SESSION['usr_datos'] = $datosArray;
//-------------------------------------------------------------------------------------------------
--------------------------
            header("Location: index.php");
        }else
            //Error cuenta desactivada estado=0
            $errormsg = "Esta cuenta esta desactivada";
    } else {
        //Error usuario o contraseña incorrectas
        $errormsg = "Revisa los datos!!!";
    }
}
```

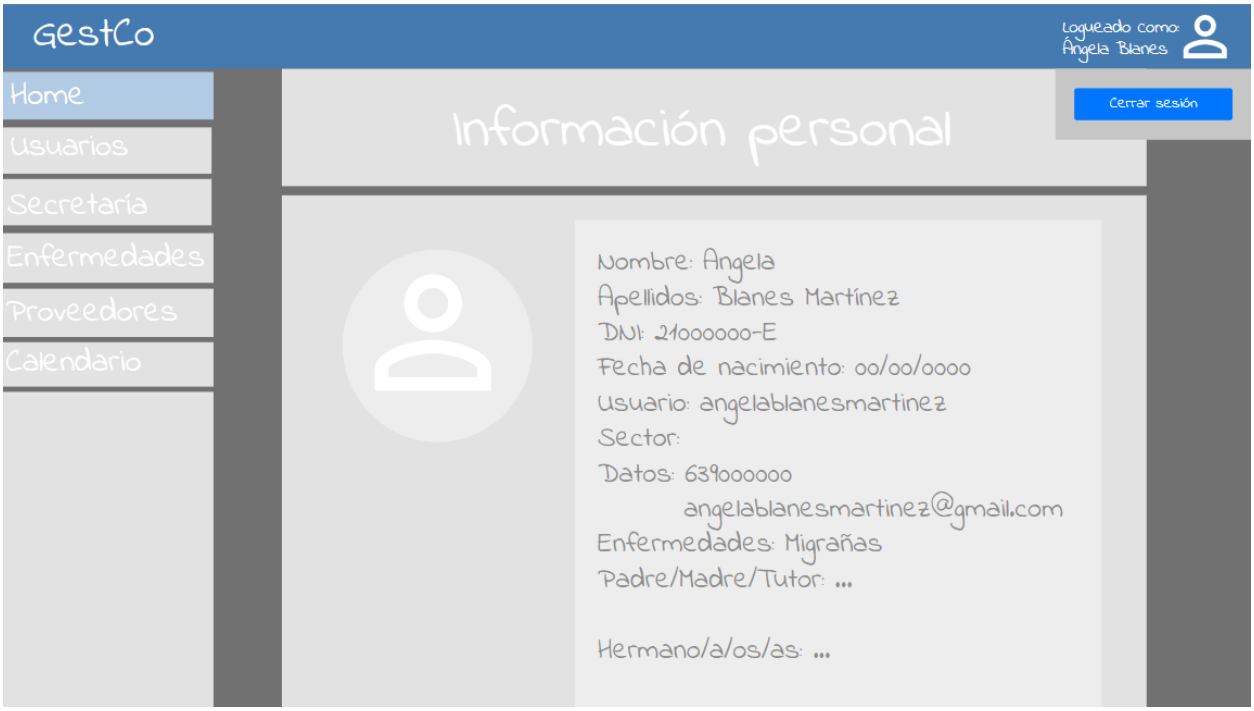

Ilustración 14. Diseño de la página principal

En la home, página que se cargará automáticamente al iniciar correctamente la sesión, se podrá ver un menú a la izquierda, en el que cargaremos todos los módulos posibles y visibles, ya que no todos los usuarios pueden verlos todos.

```
<nav id="sidebar">
   <div class="sidebar-header">
       <img src="public/img-contacto/gestco.png">
       <strong>GA</strong>
   </div>
   <div id="menu">
       <ul class="list-unstyled components">
           <li class=""><a href="?page=index"><i class="fas fa-user-
circle"></i>  <span>Información general</span></a></li>
           <?php if (isset($_SESSION['usr_id_tipo']) && ($_SESSION['usr_id_tipo'] == "1" ||
$_SESSION['usr_id_tipo'] == "6" || $_SESSION['usr_id_tipo'] == "7")) { ?>
               <li class=""><a href="?page=usuarios"><i class="fas fa-
users"></i>  <span>Usuarios</span></a></li>
               <li class=""><a href="?page=fichas_medicas"><i class="fas fa-
medkit"></i>  <span>Fichas Médicas</span></a></li>
               <li class=""><a href="?page=proveedores"><i class="fas fa-address-
book"></i>  <span>Proveedores</span></a></li>
           <?php } ?>
           <?php if (isset($_SESSION['usr_tipo']) && ($_SESSION['usr_id_tipo'] == "8" ||
$_SESSION['usr_id_tipo'] == "6" || $_SESSION['usr_id_tipo'] == "7")) { ?>
               <li class=""><a href="?page=secretaria"><i class="fas fa-
calculator"></i>  <span>Secretaria</span></a></li>
           <?php } ?>
           <li class=""><a href="?page=calendario"><i class="fas fa-
calendar"></i>  <span>Calendario</span></a></li>
       </ul>
   </div>
   <ul class="list-unstyled CTAs">
       <?php if (isset($_SESSION['usr_id'])) { ?>
```

```
            <li>Logeado como: <br><b><?php echo $_SESSION['usr_nombre'].' '.
$_SESSION['usr_apellidos']; ?></b></li><br>
            <li><a href="scripts/logout.php" class="article">Cerrar sesión</a></li>
        <?php } ?>
    </ul>
</nav>
```

Además, tendremos la información personal del usuario que haya iniciado sesión en el curso actual y que ha sido cargada en el Login en sesión, por lo tanto, solo hará falta mostrarla añadiendo la implementación de la relación de padre/madre/tutor con hijo o con hermano que a continuación podemos ver.
En este caso, si el rol del usuario es el de padre/madre/tutor, algunos datos no se mostrarán como: enfermedades, sector, y obviamente, la relación con algún padre.
Y si es de tipo hijo o monitor, mostraremos todos los datos menos la relación con hijos.

```
<?php if($_SESSION['usr_id_tipo'] == '1' || $_SESSION['usr_id_tipo'] == '2') { ?>
    <tr>
        <td><h6>ENFERMEDADES</h6></td>
        <td>
            <ul>
                <?php
                if (isset($_SESSION['usr_enfermedades'])) {
                    foreach ($_SESSION['usr_enfermedades'] as $enfermedad) { ?>
                        <li><?php echo $enfermedad ?></li>
                    <?php }
                }?>
            </ul>
        </td>
    </tr>
    <?php } ?>
    <?php if($_SESSION['usr_id_tipo'] == '1' || $_SESSION['usr_id_tipo'] == '2' ||
$_SESSION['usr_id_tipo'] == ''){ ?>
    <tr>
        <td><h6>PADRE/MADRE/TUTOR</h6></td>
        <td><?php $ids_padres = mysqli_query($con, "SELECT id_usuario_padre FROM usuario_usuario
WHERE id_usuario_nino = '".$_SESSION['usr_id']."';");
            while ($padre = mysqli_fetch_array($ids_padres)){
                $info_padre = mysqli_query($con, "SELECT * FROM usuario WHERE id =
'".$padre['id_usuario_padre']."';");
                while ($row = mysqli_fetch_array($info_padre)){ ?>
                    <ul class="list-unstyled">
                        <li><b><?php echo $row['nombre'].' '.$row['apellidos']; ?></b>
                            <ul>
                                <?php if($row['dni']) { ?>
                                    <li>DNI:  <?php echo $row['dni']; ?></li>
                                <?php } ?>
                                <?php
                                $datosPadre = mysqli_query($con, "SELECT tipo_dato, dato
from(SELECT D.id_usuario ,TD.tipo_dato, D.dato FROM datos D INNER JOIN tipo_dato TD ON TD.id =
D.id_tipo_dato) as X where X.id_usuario = '".$row['id']."'");
                                while ($dato = mysqli_fetch_array($datosPadre)){ ?>
                                    <li> <?php echo $dato['tipo_dato']?>: <?php echo
$dato['dato'] ?> </li>
                                <?php }?>
                            </ul>
                        </li>
                    </ul>
                <?php }
            } ?>
        </td>
    </tr>
```

```
   <tr>
       <td><h6>HERMANO/A/OS/AS</h6></td>
       <td><?php $ids_padres2 = mysqli_query($con, "SELECT id_usuario_padre FROM
usuario_usuario WHERE id_usuario_nino = '".$_SESSION['usr_id']."';");
           $query = "select distinct U.* from usuario U inner join usuario_usuario UU on U.id =
UU.id_usuario_nino where U.id != '".$_SESSION['usr_id']."' and (";
           $first = true;
           while ($padre = mysqli_fetch_array($ids_padres2)){
               $id_padre = $padre['id_usuario_padre'];
               if ($first){
                   $query = $query." UU.id_usuario_padre = '".$id_padre."'";
                   $first = false;
               } else {
                   $query = $query. " or UU.id_usuario_padre = '" . $id_padre . "'";
               }
           }
           $query = $query.");";
           $hermanos = mysqli_query($con, $query);
           if ($hermanos) {
               while ($hermano = mysqli_fetch_array($hermanos)) { ?>
                   <ul class="list-unstyled">
                       <li><b><?php echo $hermano['nombre'] . ' ' . $hermano['apellidos'];
?></b>
                           <ul>
                               <?php if ($hermano['dni']) { ?>
                                   <li>DNI: <?php echo $hermano['dni']; ?></li>
                               <?php } ?>
                               <?php
                               $datosHermano = mysqli_query($con, "SELECT tipo_dato, dato
from(SELECT D.id_usuario ,TD.tipo_dato, D.dato FROM datos D INNER JOIN tipo_dato TD ON TD.id =
D.id_tipo_dato) as X where X.id_usuario = '" . $hermano['id'] . "'");

                               while ($dato = mysqli_fetch_array($datosHermano)) { ?>
                                   <li> <?php echo $dato['tipo_dato'] ?>: <?php echo
$dato['dato'] ?> </li>
                               <?php }
                               $sector_hermano = mysqli_query($con, "SELECT * FROM (SELECT
USTU.id_usuario, S.sector, TU.tipo_usuario, USTU.ano FROM usuario_sector_tipo_usuario USTU
INNER JOIN sector S ON S.id = USTU.id_sector INNER JOIN tipo_usuario TU ON TU.id =
USTU.id_tipo_usuario) as X where X.id_usuario = '" . $hermano['id'] . "' and X.ano = '" .
$_SESSION['ano_actual'] . "' order by X.ano;");
                               if ($row = mysqli_fetch_array($sector_hermano)) {
                                   if ($row['tipo_usuario'] == 'monitor') { ?>
                                       <li>Sector: <?php echo $row['sector'] . ' (monitor)'
?></li>
                                   <?php } else { ?>
                                       <li>Sector: <?php echo $row['sector'] ?></li>
                                   <?php } ?>
                               <?php } ?>
                           </ul>
                       </li>
                   </ul>
               <?php }
           }?>
       </td>
   </tr>
<?php }
    if($_SESSION['usr_id_tipo'] == '3' || $_SESSION['usr_id_tipo'] == '4' ||
$_SESSION['usr_id_tipo'] == '5'){ ?>
   <tr>
       <td><h6>HIJO/S</h6></td>
       <td><?php $ids_hijos = mysqli_query($con, "SELECT id_usuario_nino FROM usuario_usuario
WHERE id_usuario_padre = '".$_SESSION['usr_id']."';");
           while ($hijo = mysqli_fetch_array($ids_hijos)){
               $info_hijo = mysqli_query($con, "SELECT * FROM usuario WHERE id =
'".$hijo['id_usuario_nino']."';");
               while ($row = mysqli_fetch_array($info_hijo)){ ?>
                   <ul class="list-unstyled">
                       <li><b><?php echo $row['nombre'].' '.$row['apellidos']; ?></b>
```

```
                            <ul>
                                <?php if($row['dni']) { ?>
                                    <li>DNI:  <?php echo $row['dni']; ?></li>
                                <?php } ?>
                                <?php
                                $datosHijo = mysqli_query($con, "SELECT tipo_dato, dato
from(SELECT D.id_usuario ,TD.tipo_dato, D.dato FROM datos D INNER JOIN tipo_dato TD ON TD.id =
D.id_tipo_dato) as X where X.id_usuario = '".$row['id']."'");
                                while ($dato = mysqli_fetch_array($datosHijo)){ ?>
                                    <li> <?php echo $dato['tipo_dato']?>: <?php echo
$dato['dato'] ?> </li>
                                <?php }
                                $sector_hijo = mysqli_query($con, "SELECT * FROM (SELECT
USTU.id_usuario, S.sector, TU.tipo_usuario, USTU.ano FROM usuario_sector_tipo_usuario USTU
INNER JOIN sector S ON S.id = USTU.id_sector INNER JOIN tipo_usuario TU ON TU.id =
USTU.id_tipo_usuario) as X where X.id_usuario = '".$row['id']."' and X.ano =
'".$_SESSION['ano_actual']."' order by X.ano;");
                                if ($row = mysqli_fetch_array($sector_hijo)) {
                                    if ($row['tipo_usuario'] == 'monitor') {?>
                                        <li>Sector: <?php echo $row['sector'].' (monitor)'
?></li>
                                    <?php } else { ?>
                                        <li>Sector: <?php echo $row['sector'] ?></li>
                                    <?php } ?>
                                <?php }?>
                            </ul>
                        </li>
                    </ul>
                <?php }
            } ?>
        </td>
    </tr>
```

Además, a la esquina superior derecha, pulsando en el usuario logueado, tendremos la posibilidad de cerrar sesión.

La siguiente pantalla serán todos los usuarios que hay en la base de datos activos, pudiéndolos filtrar por sectores con el select que tendremos justo arriba de la tabla.

```
<script type="application/javascript">
    $(function(){
        var $tabla = $('#usuarios');
        $('#selectTipo').change(function(){
          var value = $(this).val();
          value = value.replace(/ /g, "_");
          if (value){
            $('tbody tr.' + value, $tabla).show();
            $('tbody tr:not(.' + value + ')', $tabla).hide();
          }
           else{
            // Se ha seleccionado All
            $('tbody tr', $tabla).show();
          }
       });
    });
</script>
<div class="text-right">
    <span>SECTORES: </span>
    <select id="selectTipo">
        <option value="">TODOS</option>
        <?php $sectores = mysqli_query($con, "SELECT * FROM sector");
        while ($sector = mysqli_fetch_array($sectores)){ ?>
            <option value="<?php echo $sector['sector']; ?>"><?php echo $sector['sector'];
?></option>
```

```
        <?php }?>
    </select>
</div><br>
```

Además, podrán añadirse usuarios nuevos pulsando en el botón verde y podremos ver con detalle cualquier usuario, editarlo o eliminarlo, que, en este caso, lo desactivaremos.

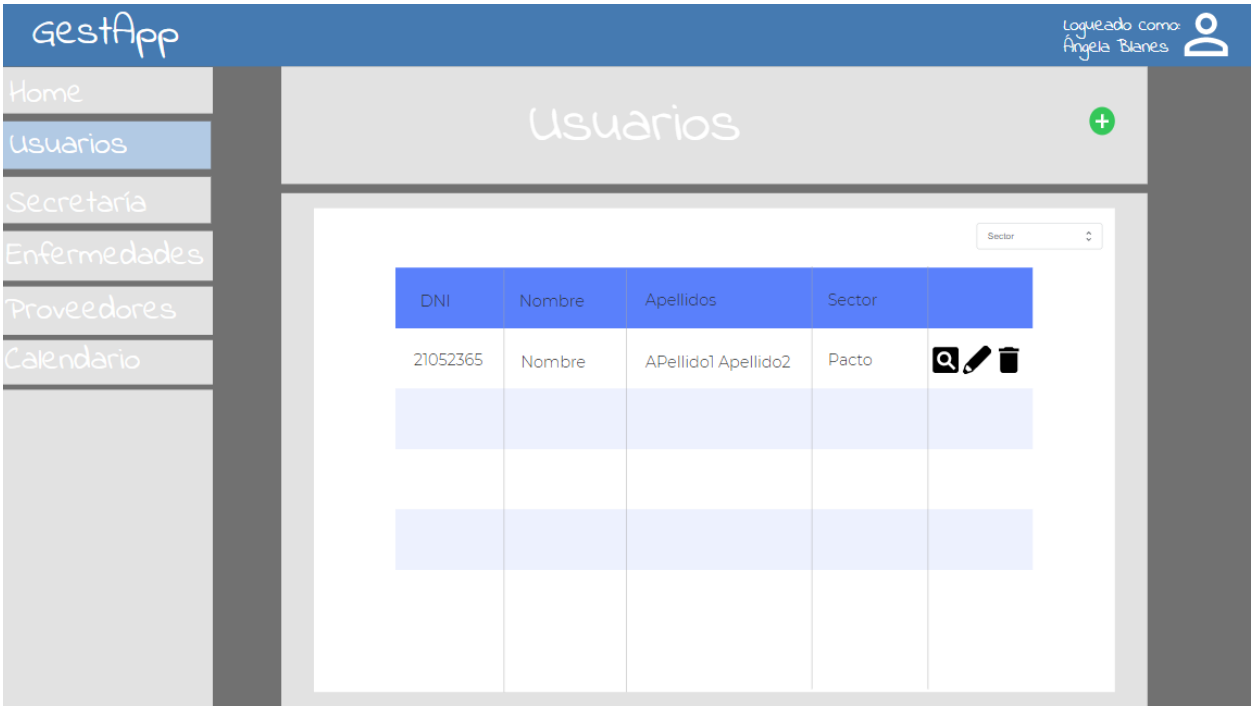

Ilustración 15. Diseño de Usuarios

El botón de eliminar un usuario, simplemente hará un Update de la bbdd poniendo el estado_activo del usuario a 0.

Y tanto para editarlo como para ver un usuario, tendremos un modal popup que aparecerá superponiéndose y con el que haremos un Update dependiendo de si estamos viéndolo o editándolo.

```
<form action="view/scripts/guardarUsuarioBD.php" method="post" name="formCrearUsuario"
id="formCrearUsuario">
    <table class="table">
        <?php $tipos = mysqli_query($con, "SELECT * FROM tipo_usuario"); ?>
        <tr>
            <td>Tipo de usuario: </td>
            <td>
                <form action="./" method="post">
                    <select id="status" name="status" onChange="mostrar(this.value);" required>
                        <option value=""></option>
                        <?php while ($tipo = mysqli_fetch_array($tipos)){ ?>
                            <option value="<?php echo $tipo['id']; ?>"><?php echo
$tipo['tipo_usuario']; ?></option>
                        <?php } ?>
```

```
                    </select>
                </form>
            </td>
        </tr>
        <tr id="dni">
            <td>DNI: </td>
            <td><input type="text" name="dni" id="dni" required/></td>
        </tr>
        <tr id="nombre">
            <td>Nombre: </td>
            <td><input type="text" name="nombre" id="nombre" autofocus required/></td>
        </tr>
        <tr id="apellidos">
            <td>Apellidos: </td>
            <td><input type="text" name="apellidos" id="apellidos" required/></td>
        </tr>
        <tr id="sector" style="display: none;">
            <td>Sector: </td>
            <td>
                <select id="selectSector" name="sector">
                    <option value=""></option>
                    <?php $sectores = mysqli_query($con, "SELECT * FROM sector");
                    while ($sector = mysqli_fetch_array($sectores)){ ?>
                        <option value="<?php echo $sector['id']; ?>"><?php echo
$sector['sector']; ?></option>
                    <?php }?>
                </select>
            </td>
        </tr>
        <tr id="password">
            <td>Password: </td>
            <td><input type="password" name="password" id="password" required/></td>
        </tr>
        <tr id="fecha_nacimiento" style="display: none;">
            <td>Fecha de nacimiento: </td>
            <td><input type="date" name="fecha_nacimiento" id="fecha_nacimiento"></td>
        </tr>
        <tr id="datos">
            <td>Dato/s: </td>
            <td>
                <table class="table-borderless">
                    <tr id="datoOriginal" hidden>
                        <td>
                            <select>
                                <option value=""></option>
                                <?php $tiposDatos = mysqli_query($con, "SELECT * FROM
tipo_dato");
                                while ($tipoDato = mysqli_fetch_array($tiposDatos)){ ?>
                                    <option value="<?php echo $tipoDato['id']; ?>"><?php echo
$tipoDato['tipo_dato']; ?></option>
                                <?php }?>
                            </select>
                        </td>
                        <td>
                            <input type="text"/>
                        </td>
                        <td>
                            <input type="button" id="borrar" class="borrar btn-outline-danger"
value="X" />
                        </td>
                    </tr>
                    <div id="datoClonado"></div>
                    <tr>
                        <td>
                            <select name="selectTipoDato[]">
                                <option value=""></option>
                                <?php $tiposDatos = mysqli_query($con, "SELECT * FROM
tipo_dato");
                                while ($tipoDato = mysqli_fetch_array($tiposDatos)){ ?>
```

```
                                    <option value="<?php echo $tipoDato['id']; ?>"><?php echo
$tipoDato['tipo_dato']; ?></option>
                                <?php }?>
                            </select>
                        </td>
                        <td>
                            <input type="text" name="dato[]"/>
                        </td>
                        <td>
                            <input type="button" class="anadir btn-outline-primary" value="+" />
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr id="enfermedades" style="display:none;">
            <td>Enfermedades: </td>
            <td>
                <select name="selectEnfermedad[]" id="selectEnfermedad" multiple>
                    <option value="" selected></option>
                    <?php $enfermedades = mysqli_query($con, "SELECT * FROM enfermedad");
                    while ($enfermedad = mysqli_fetch_array($enfermedades)){ ?>
                        <option value="<?php echo $enfermedad['id']; ?>"><?php echo
$enfermedad['nombre']; ?></option>
                    <?php }?>
                </select>
            </td>
        </tr>
        <tr id="padres" style="display:none;">
            <td>Padre/Madre/Tutor: </td>
            <td>
                <select name="selectPadre[]" id="selectPadre" multiple>
                    <option value="" selected></option>
                    <?php $padres = mysqli_query($con, "select distinct U.id, U.nombre,
U.apellidos
                                                        from usuario U
                                                        inner join
usuario_sector_tipo_usuario USTU
                                                        on U.id = USTU.id_usuario
                                                        where USTU.id_tipo_usuario = '3'
|| USTU.id_tipo_usuario = '4' || USTU.id_tipo_usuario = '5';");
                    while ($padre = mysqli_fetch_array($padres)){ ?>
                        <option value="<?php echo $padre['id']; ?>"><?php echo
$padre['nombre'].' '.$padre['apellidos']; ?></option>
                    <?php }?>
                </select>
            </td>
        </tr>
    </table>
    <input type="submit" name="submit" value="Añadir usuario">
    <input type="reset" name="clear" value="Borrar">
</form>
```

Para ocultar o mostrar los diferentes campos requeridos, se rige dependiendo del tipo de usuario que sea, como se ve a continuación:

```
<script type="text/javascript">
    function mostrar(id) {
    if (id == "") {
        $("#sector").hide();
        $("#fecha_nacimiento").hide();
        $("#enfermedades").hide();
        $("#padres").hide();
    }
    if (id == "1" || id == "2") {
```

```
        $("#sector").show();
        $("#fecha_nacimiento").show();
        $("#enfermedades").show();
        $("#padres").show();
    }
    if (id == "3" || id == "4" || id == "5") {
        $("#sector").hide();
        $("#fecha_nacimiento").hide();
        $("#enfermedades").hide();
        $("#padres").hide();
    }
    if (id == "7" || id == "6" || id == "8") {
        $("#sector").hide();
        $("#fecha_nacimiento").hide();
        $("#enfermedades").hide();
        $("#padres").hide();
    }
}
</script>
```

Al dar de alta un nuevo usuario, nos aparecerá el siguiente modal… y será el mismo que a la hora de editarlo con la diferencia de que este no cargará ningún valor y será todo de cero.

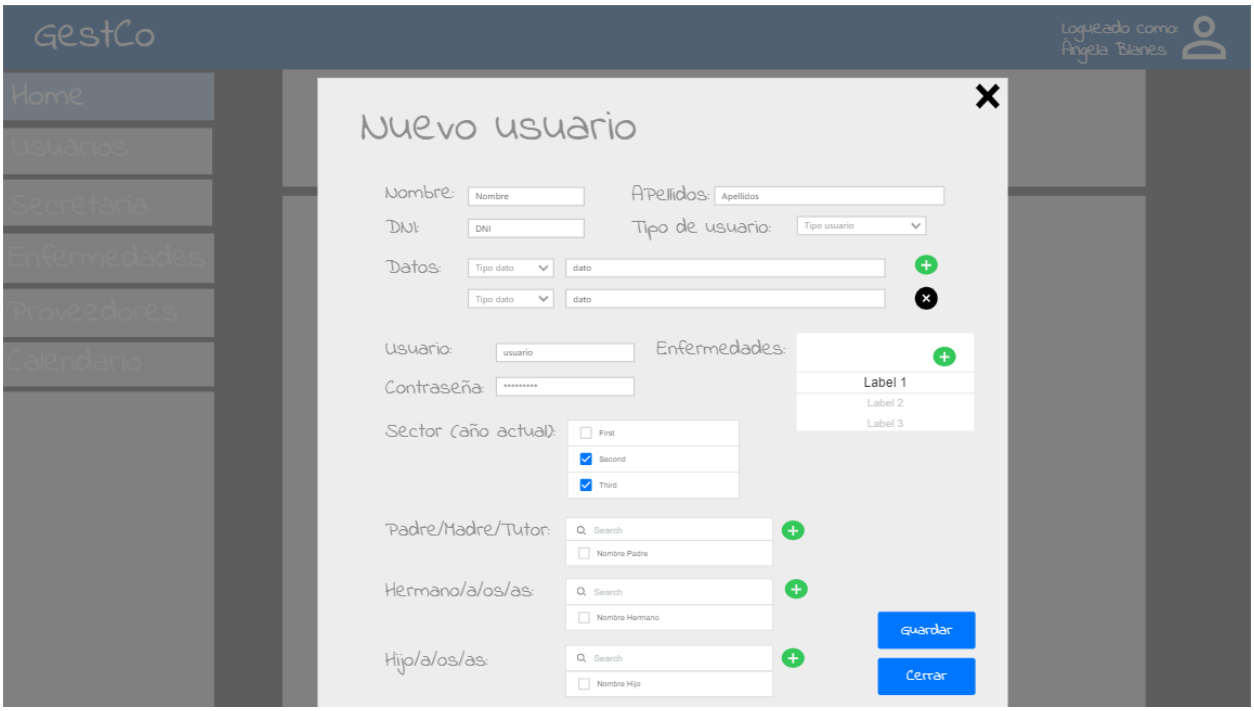

Ilustración 16. Diseño del formulario para dar de alta a un usuario

Otra de las pestañas sería Secretaría, en la que se podrán ver los pagos que existen y a la derecha, una lista de usuarios para seleccionar si han pagado o no. Además, se podrán dar de alta pagos.

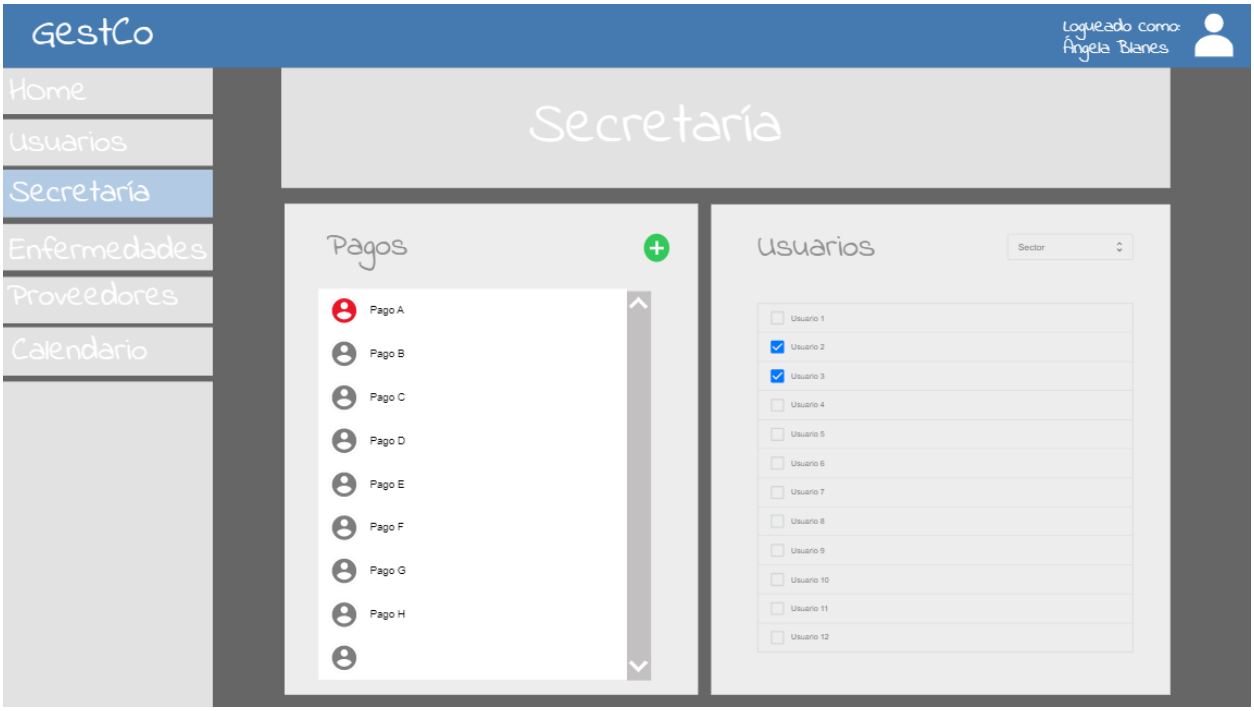

Ilustración 17. Diseño de Secretaría

En este caso, la relación del pago con el usuario se hace de la siguiente manera:

```
<table class="table table-striped thead-dark" id="pagos">
   <thead>
   <tr>
       <th>Actividad</th>
       <th>Precio</th>
       <th></th>
   </tr>
   </thead>
   <tbody>
   <?php $pagos = mysqli_query($con, "SELECT * FROM pagos where ano =
'".$_SESSION['ano_actual']."';");
   while ($pago = mysqli_fetch_array($pagos)){ ?>
       <tr value="<?php echo $pago['id']; ?>">
           <td><?php echo $pago['nombre']; ?></td>
           <td><?php echo $pago['cantidad']; ?> €</td>
           <td>
               <button type="button" class="btn btn-primary" data-toggle="modal" data-
target="#editarPago" onclick="buscarUsuariosPago(<?php echo $pago['id']; ?>,'<?php echo
$pago['nombre']; ?>')"><i class="fas fa-edit"></i></button>
               <button class="btn btn-danger" data-toggle="modal" data-target="#eliminarPago"
onclick="borrarActividad(<?php echo $pago['id'];?>)"><i class="fas fa-trash"></i></button>
           </td>
       </tr>
   <?php }?>
   </tbody>
</table>
```

Con la función buscarUsuariosPagos, se marcan aquellos usuarios del pago seleccionado que hayan pagado pasándole el id del pago.

```
function buscarUsuariosPago(idPago, nombrePago) {
   $('input:checkbox').removeAttr('checked');
   $('#usuariosNumPago').text(idPago);
   $('#nombreActividad').text(nombrePago);

   $.post("view/scripts/buscarUsuariosPagados.php", {id: idPago}, function(datos){
       ids = datos.split(';');
       console.log(ids);
       var lenghtIds = ids.length;
       for (var i = 0; i<lenghtIds ;i++){
           $("#"+ids[i]).prop('checked', 'true');
       }
   });
   $('#formularioUsuariosPago').refresh();
};
```

Además, se podrá actualizar la lista de usuarios que hayan pagado agregándose los usuarios marcados al pago.

```
$(document).ready(function() {
   $('#actualizar').click(function(){
   var selected = '';
   $('#formularioUsuariosPago input[type=checkbox]').each(function(){
       if (this.checked) {
           selected += $(this).val()+',';
       }
   });
   pago = $('#usuariosNumPago').text();
   if (selected != ''){
       seleccionados=selected.split(',');
       seleccionados.pop();
       $.post("view/scripts/insertarUsuariosPagados.php",{"arraySeleccionados":seleccionados,
"pago":pago},function(respuesta){
           alert(respuesta);
       });
   }
   else{
       alert('Debes seleccionar al menos una opción.');
   }
   return false;
});

});
```

En 'Enfermedades', podremos ver una lista de usuarios que padezcan alguna enfermedad/alergia y filtrarlos por sector para verlo mejor, utilizando el mismo select que se ha implementado con los usuarios.

Ilustración 18. Diseño de Enfermedades

En Proveedores podremos dar de alta nuevos proveedores con un formulario muy parecido al de los usuarios de antes y será una lista con información (dirección, teléfono…) de diferentes lugares como albergues, sitios donde comprar comida al por mayor… pudiendo filtrar por el tipo.

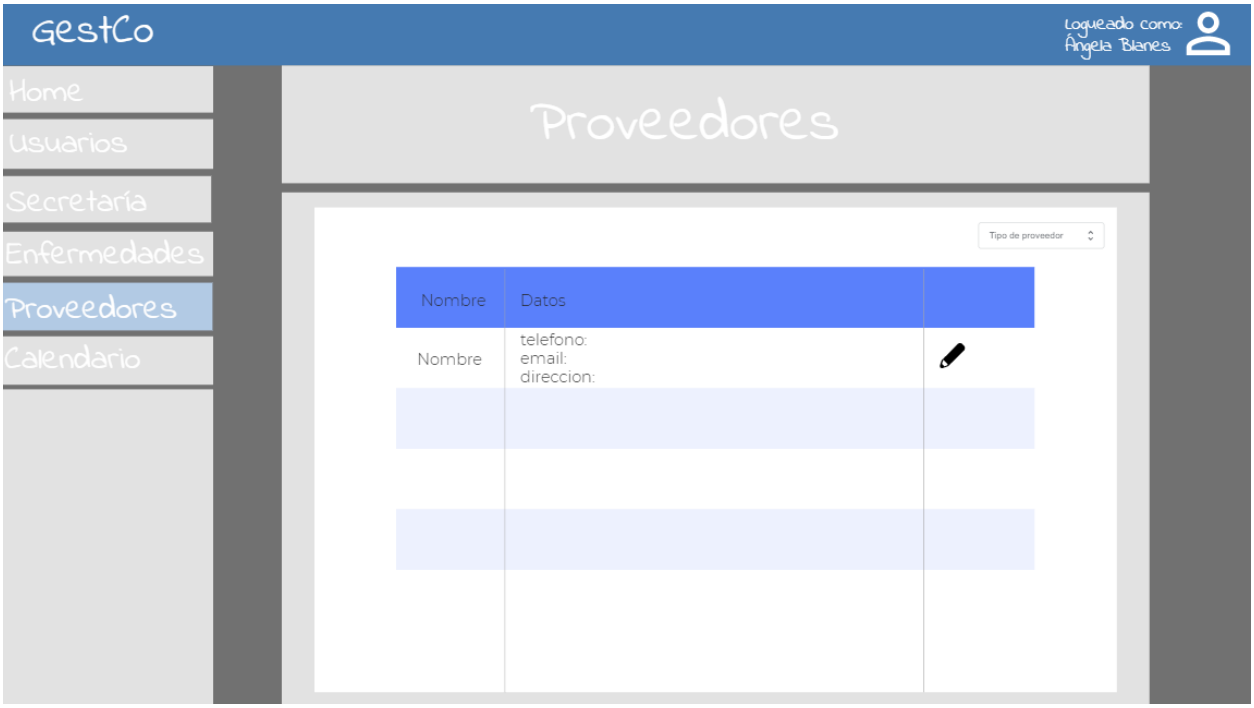

Ilustración 19. Diseño de Proveedores

Se ha tenido en cuenta a la hora de introducir de datos, que pueden ser varios, por lo que se van añadiendo y luego se introducen en la base de datos como registros separados.

El modal sería el siguiente:

```
<div class="modal fade bd-example-modal-lg" id="crearProveedor" tabindex="-1" role="dialog"
aria-labelledby="myLargeModalLabel" aria-hidden="true">
   <div class="modal-dialog modal-lg">
       <div class="modal-content">
           <div class="modal-header">
               <h5 class="modal-title" id="exampleModalLongTitle">Nuevo proveedor</h5>
               <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                   <span aria-hidden="true">&times;</span>
               </button>
           </div>
           <div class="modal-body">
               <form action="view/scripts/guardarProveedorBD.php" method="post"
name="formCrearProveedor" id="formCrearProveedor">
                   <table class="table">
                       <tr>
                           <td>Nombre: </td>
                           <td><input type="text" name="nombre" id="nombre" autofocus
required/></td>
                       </tr>
                       <tr>
                           <td>Dato/s: </td>
                           <td>
                               <table class="table-borderless">
                                   <tr id="datoOriginal" hidden>
                                       <td>
                                           <select>
                                               <option value=""></option>
                                               <?php $tiposDatos = mysqli_query($con, "SELECT *
FROM tipo_dato");
                                               while ($tipoDato =
mysqli_fetch_array($tiposDatos)){ ?>
                                                   <option value="<?php echo $tipoDato['id'];
?>"><?php echo $tipoDato['tipo_dato']; ?></option>
                                               <?php }?>
                                           </select>
                                       </td>
                                       <td>
                                           <input type="text"/>
                                       </td>
                                       <td>
                                           <input type="button" id="borrar" class="borrar btn-
outline-danger" value="X" />
                                       </td>
                                   </tr>
                                   <div id="datoClonado"></div>
                                   <tr>
                                       <td>
                                           <select name="selectTipoDato[]">
                                               <option value=""></option>
                                               <?php $tiposDatos = mysqli_query($con, "SELECT *
FROM tipo_dato");
                                               while ($tipoDato =
mysqli_fetch_array($tiposDatos)){ ?>
                                                   <option value="<?php echo $tipoDato['id'];
?>"><?php echo $tipoDato['tipo_dato']; ?></option>
                                               <?php }?>
                                           </select>
                                       </td>
                                       <td>
                                           <input type="text" name="dato[]"/>
```

```
                                    </td>
                                    <td>
                                        <input type="button" id="anadir" class="anadir btn-
outline-primary" value="+" style="display: block;"/>
                                        <input type="button" id="borrar" class="borrar btn-
outline-danger" value="X" style="display: none;"/>
                                    </td>
                                </tr>
                            </table>
                            <script type="application/javascript">
                                $(function () {
                                    $(document).on('click', '.borrar', function (event) {
                                        event.preventDefault();
                                        $(this).closest('tr').remove();
                                    });
                                    $(document).on('click', '.anadir', function (event) {
                                        var clone =
$('#datoOriginal').clone().appendTo('#datoClonado');
                                        clone.removeAttr('hidden');
                                        clone.removeAttr('id');
                                        clone.find("select").attr("name",
"selectTipoDato[]");
                                        clone.find("input").attr("name", "dato[]");
                                    });
                                });
                            </script>
                        </td>
                    </tr>
                    <tr valign="top">
                        <td>Tipo de proveedor: </td>
                        <td>
                            <select name="selectTipo[]" id="selectTipo" multiple>
                                <?php $tipos = mysqli_query($con, "SELECT * FROM
tipo_proveedor");
                                while ($tipo = mysqli_fetch_array($tipos)){ ?>
                                    <option value="<?php echo $tipo['id']; ?>"><?php echo
$tipo['tipo_proveedor']; ?></option>
                                <?php }?>
                            </select>
                        </td>
                    </tr>
                </table>
                <input type="submit" name="submit" value="Dar de alta proveedor">
                <input type="reset" name="clear" value="Borrar">
            </form>
        </div>
    </div>
</div>
```

Y, por último, el diseño del calendario, será un calendario en el que podremos visualizar todas las actividades diferenciadas por el color dependiendo del sector.

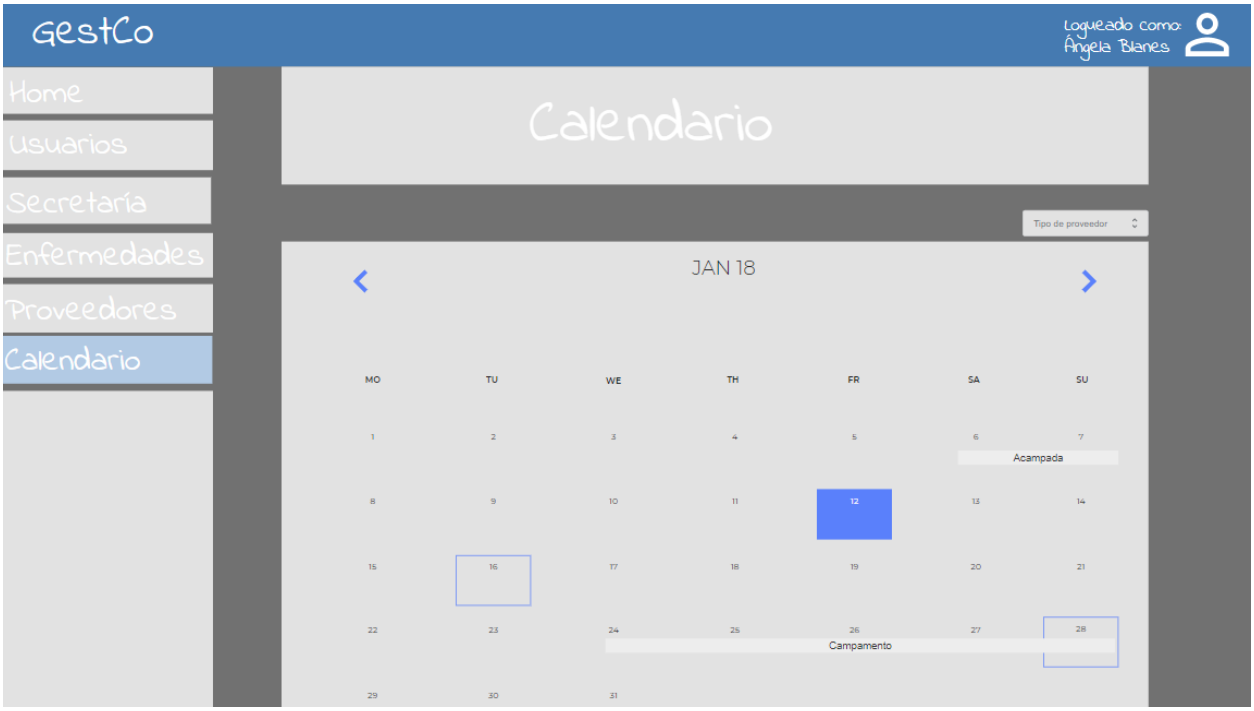

Ilustración 20. Diseño de Calendario

Para crear el calendario, se ha utilizado un plugin jQuery que se llama fullcalendar.io y se ha modificado para que se adapte a nuestras necesidades.
Para añadir una actividad, tendremos un modal que nos pide: el nombre, el sector (para darle un color diferente), la fecha de inicio y la fecha final. Y para editar, solo se podrá el sector y el nombre.

Habrá dos ficheros:
añadirActividad

```
if (isset($_REQUEST['title']) && isset($_REQUEST['color']) && isset($_REQUEST['start']) &&
isset($_REQUEST['end'])){

  $title = $_REQUEST['title'];
  $color = $_REQUEST['color'];
  $start = $_REQUEST['start'];
  $end = $_REQUEST['end'];


  $query = "INSERT INTO actividad(title, start, end, color) values ('$title', '$start', '$end',
'$color');";
  error_log($query);
  if($sql = mysqli_query($con, $query)){
     echo 'Actividad insertada';
  }
}
```

editarTituloActividad, que también sirve para eliminar la actividad:

```php
if (isset($_REQUEST['delete']) && isset($_REQUEST['id'])){
  $id = $_REQUEST['id'];
  $sql = "DELETE FROM actividad WHERE id = $id";
  $query = $con->prepare( $sql );
  if ($query == false) {
   print_r($con->errorInfo());
   die ('Erreur prepare');
  }
  $res = $query->execute();
  if ($res == false) {
   print_r($query->errorInfo());
   die ('Erreur execute');
  }

}elseif (isset($_REQUEST['title']) && isset($_REQUEST['color']) && isset($_REQUEST['id'])){

  $id = $_REQUEST['id'];
  $title = $_REQUEST['title'];
  $color = $_REQUEST['color'];

  $sql = "UPDATE actividad SET  title = '$title', color = '$color' WHERE id = $id ";


  $query = $con->prepare( $sql );
  if ($query == false) {
   print_r($con->errorInfo());
   die ('Erreur prepare');
  }
  $sth = $query->execute();
  if ($sth == false) {
   print_r($query->errorInfo());
   die ('Erreur execute');
  }
}
```

3.2.3 Diagramas de funcionamiento

Y, por último, tendríamos la capa 2, es decir, la lógica de negocio, donde vemos cuál será el funcionamiento de la aplicación y el flujo entre las diferentes pantallas.
Primero que nada, podemos ver el diagrama de flujo de lo que sería el inicio de sesión en la aplicación.

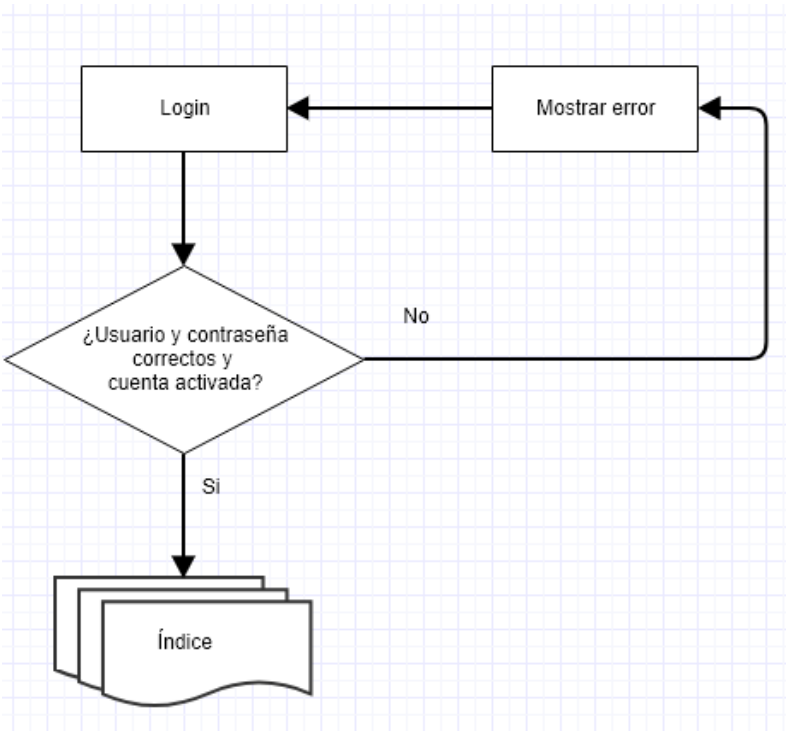

Ilustración 21. Diagrama del flujo del Login

Y otro diagrama de flujo, a la hora de dar de alta un usuario o proveedor.

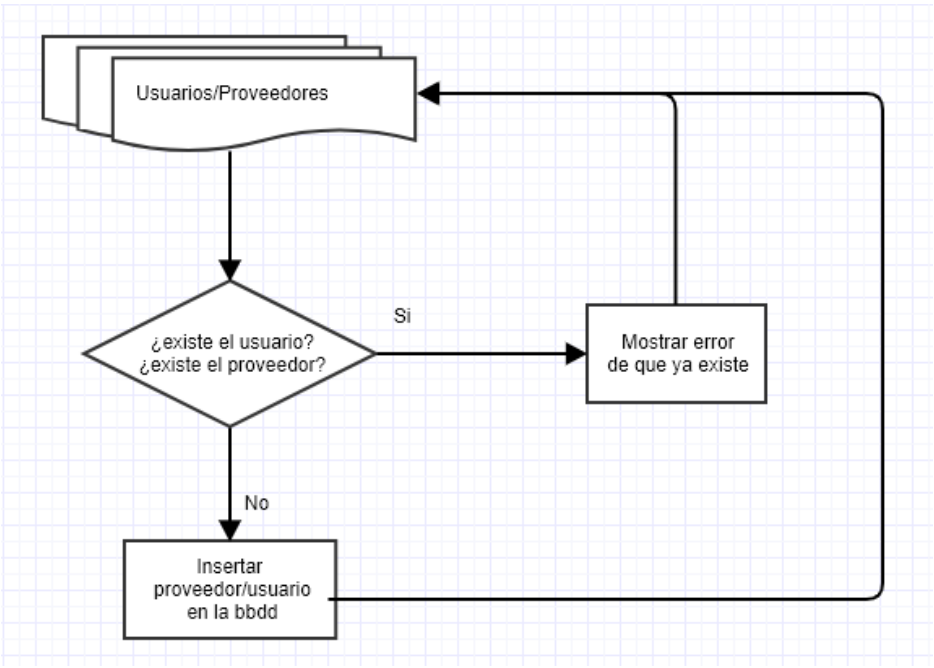

Ilustración 22. Diagrama del flujo al dar de alta un usuario

Una vez iniciada sesión, dependiendo del tipo de usuario que se sea, se podrá ver una serie de módulos distintos que detallo también a continuación con unos diagramas de casos de uso y las tablas que podemos ver a continuación y donde se especifica las secciones que se verán desde cada usuario en el menú y la información que cada usuario verá en el inicio.

En esta tabla, se ve el tipo de usuario en la primera columna, el usuario (DNI) que se ha utilizado para las pruebas y las secciones del menú lateral de la izquierda que verán cada tipo.

password	123456	VISUALIZACIÓN	ACCESOS					
	usuario	Menu	Info general	Usuarios	Fichas medicas	Proveedores	Secretaría	Calendario
Jefatura	21000001	todo	X	X	X	X	X	X
admin	21000002	todo	X	X	X	X	X	X
secretaria	21000003	secretaria	X				X	X
monitor	21000004	todo menos secretaria	X	X	X	X		X
padre	21000005	info y calendario	X					X
madre	21000006	info y calendario	X					X
tutor	21000007	info y calendario	X					X
niño	21000008	info y calendario	X					X
niño2	21000009	info y calendario	X					X

Y en la siguiente, se puede ver que campos verán en la página de su información personal porque dependiendo del tipo, nos interesa mostrar unas u otras.

		INFO GENERAL									
	usuario	Nombre	Apellidos	DNI	F.NACIMIENTO	SECTOR	DATOS	ENFERMEDADES	PADRE/MADRE/TUTOR	HERMANOS	HIJOS
Jefatura	21000001	X	X	X			X				
admin	21000002	X	X	X			X				
secretaria	21000003	X	X	X			X				
monitor	21000004	X	X	X	X	X	X	X	X	X	
padre	21000005	X	X	X			X				X
madre	21000006	X	X	X			X				x
tutor	21000007	X	X	X			X				x
niño	21000008	X	X	X	X	X	X	X	X	X	
niño2	21000009	X	X	X	X	X	X	X	X	X	

A continuación, se pasan a detallar los diagramas de casos de uso.

- Un usuario del tipo 'monitor'

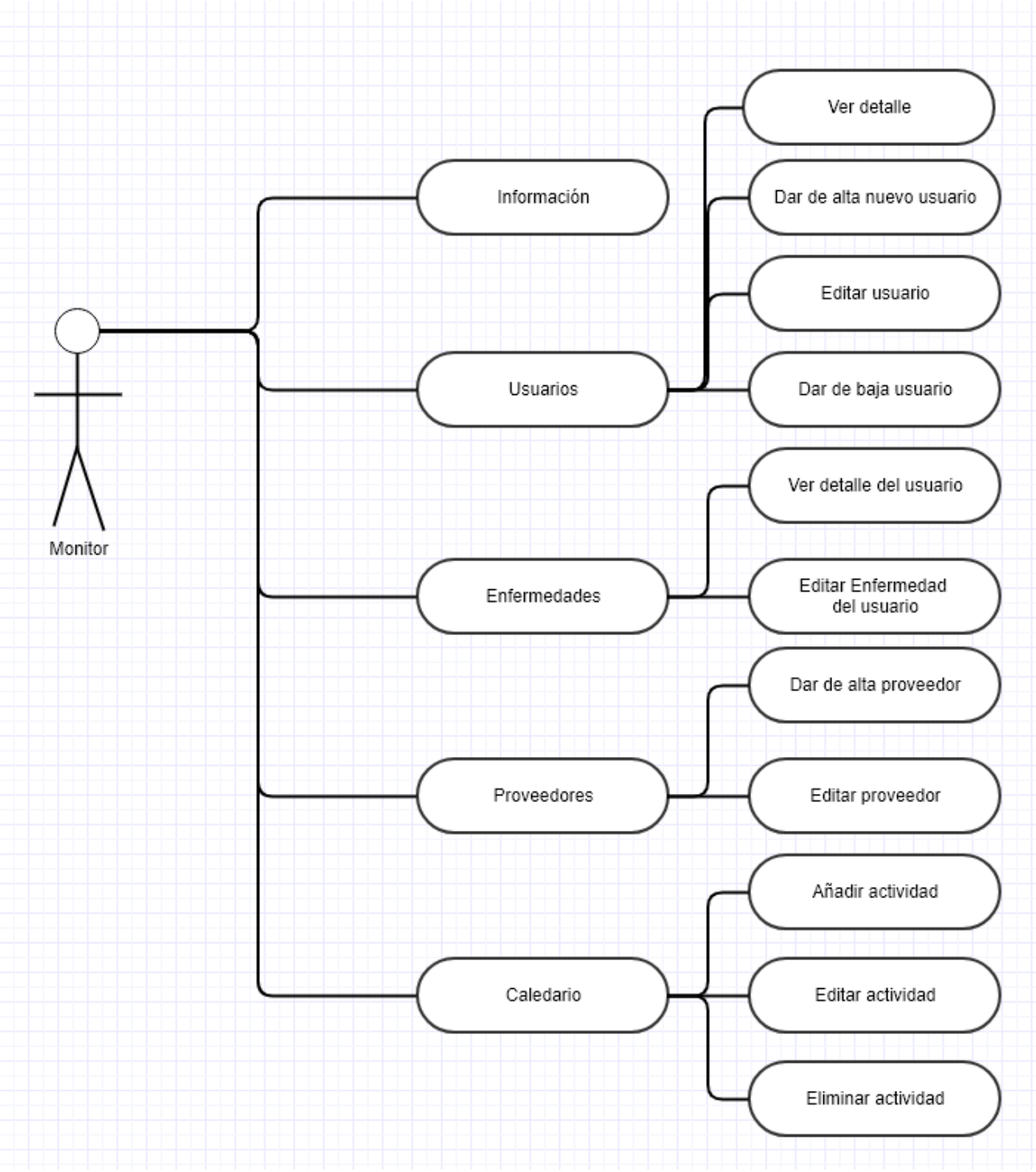

Ilustración 23. Caso de uso Monitor

- Un usuario del tipo 'jefatura' o 'admin'

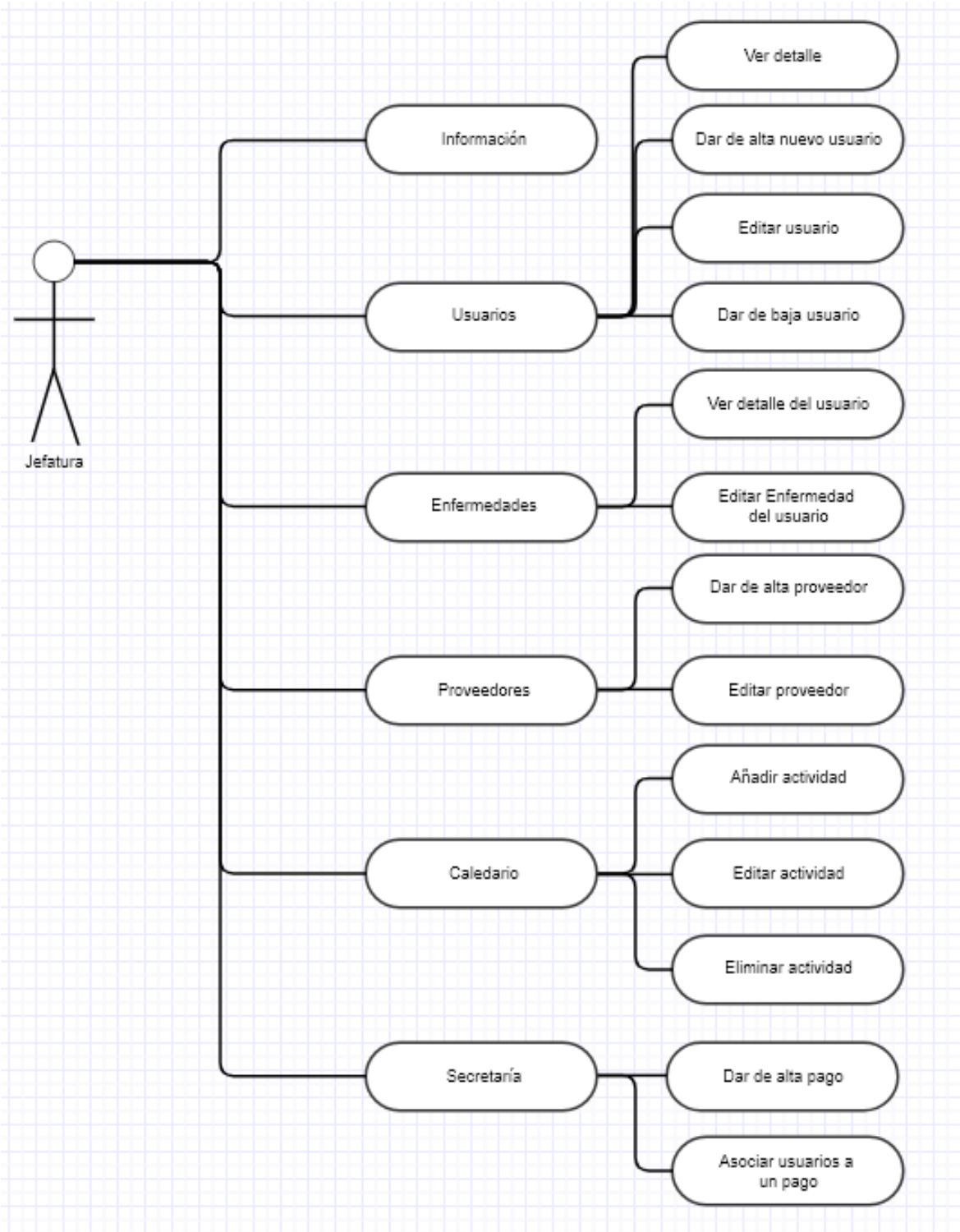

Ilustración 24. Caso de uso Jefatura

- Un usuario del tipo 'secretaria'

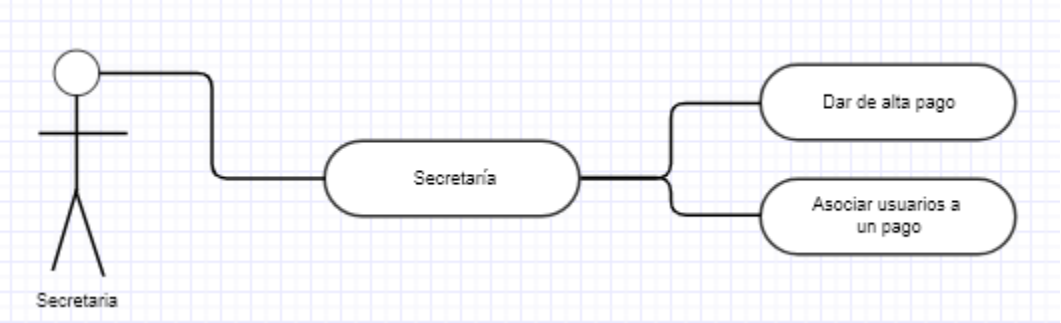

Ilustración 25. Caso de uso Secretario

- Un usuario del tipo 'niño' o 'padre'

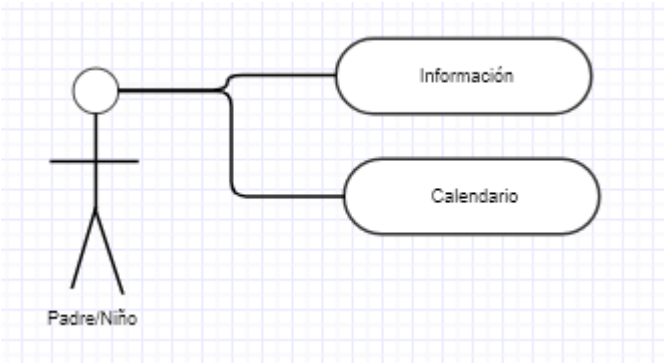

Ilustración 26. Caso de uso Padre o Niño

3.3. Pruebas

En el siguiente punto, buscamos el funcionamiento correcto de la aplicación, y para ello se hacen una serie de pruebas en las que debemos detectar si existiera algún error y poder resolverlo, por lo que son muy importantes a la hora de desarrollar cualquier aplicación.

3.3.1 De sistema

Primero he realizado pruebas de sistema,

- **Validación de campos obligatorios vacíos**
  En este caso, se ha comprobado que cuando un campo es obligatorio en la bbdd, siempre que se vaya a guardar un nuevo registro, no nos deje hacerlo si está vacío. En el caso de hacerlo y que el campo esté vacío, nos dará un aviso como el siguiente.

Ilustración 27. Prueba de un campo requerido

- **Validación de que se actualizan o se insertan los datos en la base de datos al hacer cualquier modificación o dar de alta un nuevo usuario o proveedor.**

- **Y se ha validado que la aplicación y todas sus secciones/módulos se vea correctamente en varios navegadores** (Chrome, Firefox, Edge…)

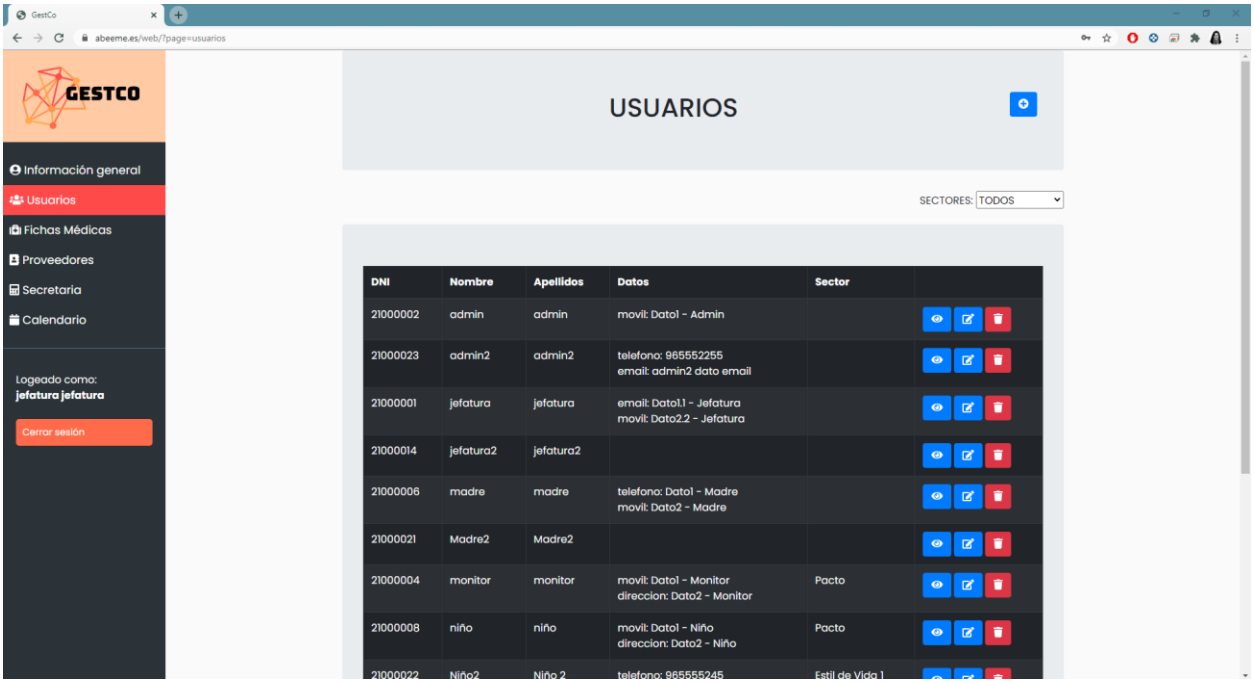

Ilustración 28. Prueba en Chrome

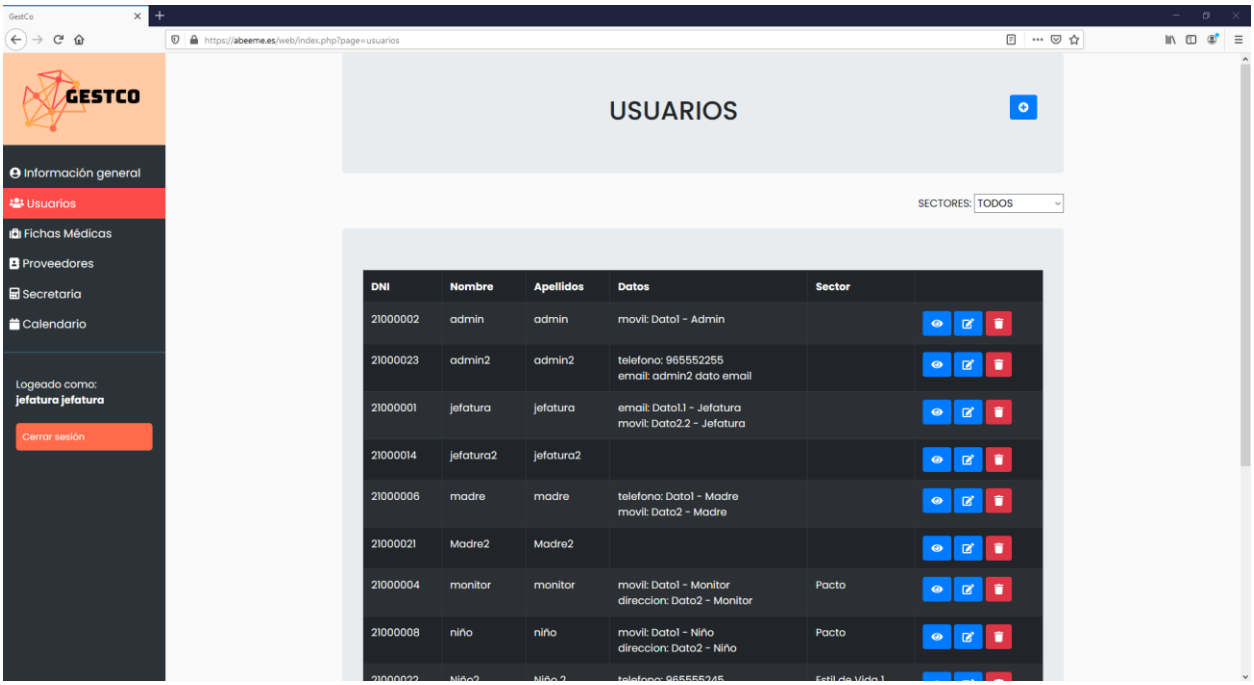

Ilustración 29. Prueba en Firefox

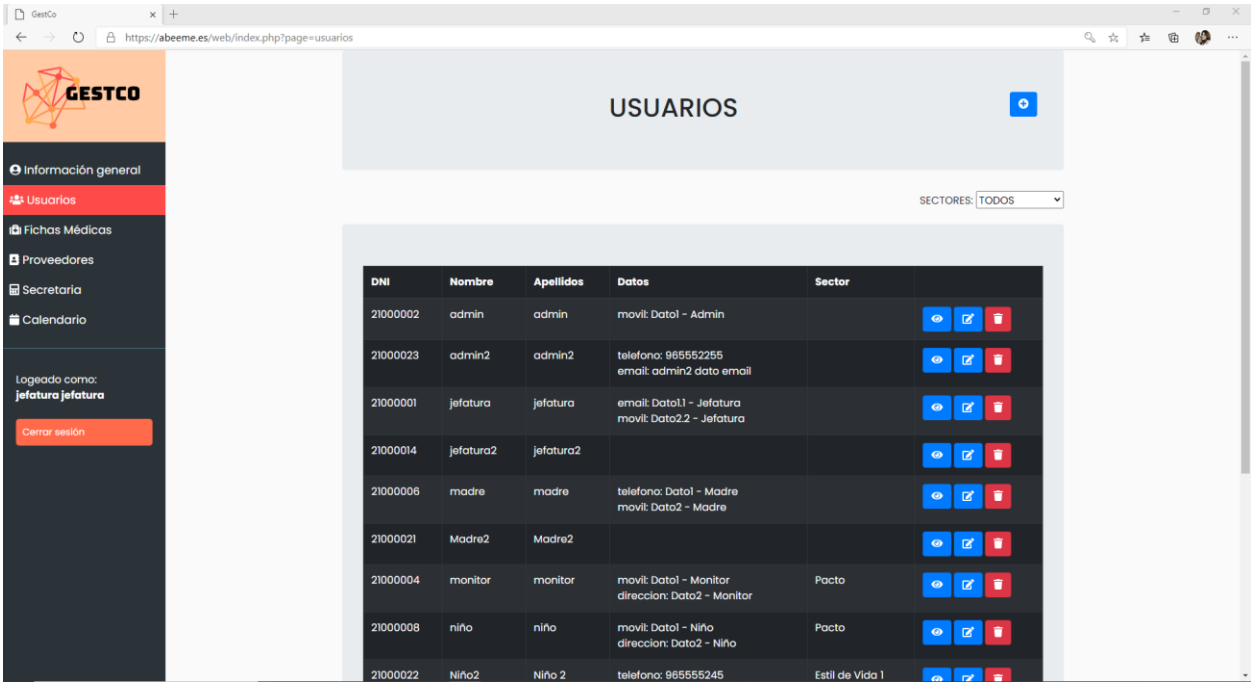

Ilustración 30. Prueba en Edge

3.3.2 De integración de sistemas

Estas pruebas son aquellas que las realizamos una vez está el módulo o sección desarrollado.

- **Comprobar que la sesión no se cierra**
  Se ha probado que hasta que el usuario no cierra la sesión, la sesión sigue abierta incluso si se cierra el navegador y se vuelve a abrir.

- **Permisos de los usuarios**
  A la hora de iniciar sesión, dependiendo del tipo de usuario que se sea, se mostrarán ciertas secciones.
  - ❏ En el caso de 'secretario', solo 'Secretaría', 'Información personal' y 'Calendario'.
  - ❏ En el caso de 'niño' o 'padre/madre/tutor', solo la información personal suya y el calendario.
  - ❏ En el caso de 'monitor', todas menos secretaría.
  - ❏ Y en el resto (jefatura, admin), todas.

- **Puesta en marcha en producción**
  Se ha probado, una vez subida la aplicación al hosting, que todo funcionara bien.
  https://abeeme.es/

3.3.3 De volumen

Son aquellas pruebas en las que debemos poner a nuestra aplicación a toda máquina, es decir, ponernos en la situación que más recursos se estén consumiendo, y en este caso, solo se ha podido hacer una ya que la prueba de volumen con el máximo número de registros en la bbdd es casi imposible porque en el centro en el que se va a implantar hay unos 500 niños, con sus respectivos padres… y sería introducir una base de datos con más de 1000 registros solamente de usuarios.

La prueba que sí se ha realizado ha sido la de acceder con el máximo número de usuarios que se pueden dar en una posible situación, que, en el caso del centro, sería en un pago de campamento en el que habría un monitor por sector conectado a la misma aplicación, es decir, 7 monitores como máximo conectados a la vez y no ha dado problema alguno.
Para ello, se han gastado 2 equipos y en cada equipo, 3 navegadores diferentes más una Tablet para llegar a los 7 usuarios.

4 Resultados

4.1 Migración al entorno de producción

Para la migración de la aplicación web a un entorno de producción, se ha optado por contratar un hosting e instalar la aplicación allí ya que ahora mismo en el centro no tenemos los recursos suficientes como para poder alojarlo allí mismo.

La migración no es para nada complicada ya que, primero que nada, debemos ver dónde contratar el hosting, y cuando lo tenemos, activar el dominio verificándolo primero para ver si existe. Esto tarda entre 24 y 48 horas en activarse.

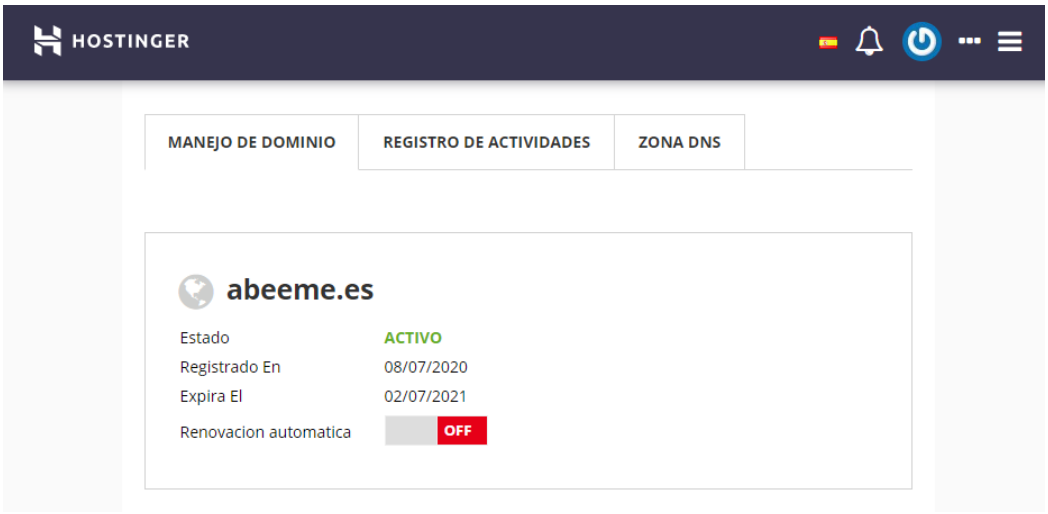

Ilustración 31. Registro hosting

Después, daremos de alta un hosting y podremos crear la base de datos que vayamos a utilizar, exportando nuestra base de datos local para importarla directamente con un script.

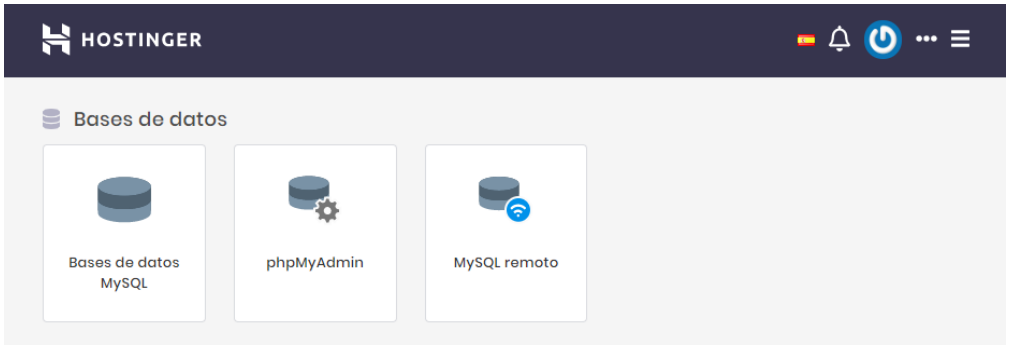

Ilustración 32. Configuración bases de datos en el hosting

Desde Bases de datos MySQL, daremos de alta la base de datos y después, por phpmyadmin accederemos para hacer el import del export que hemos hecho de nuestra base de datos local.

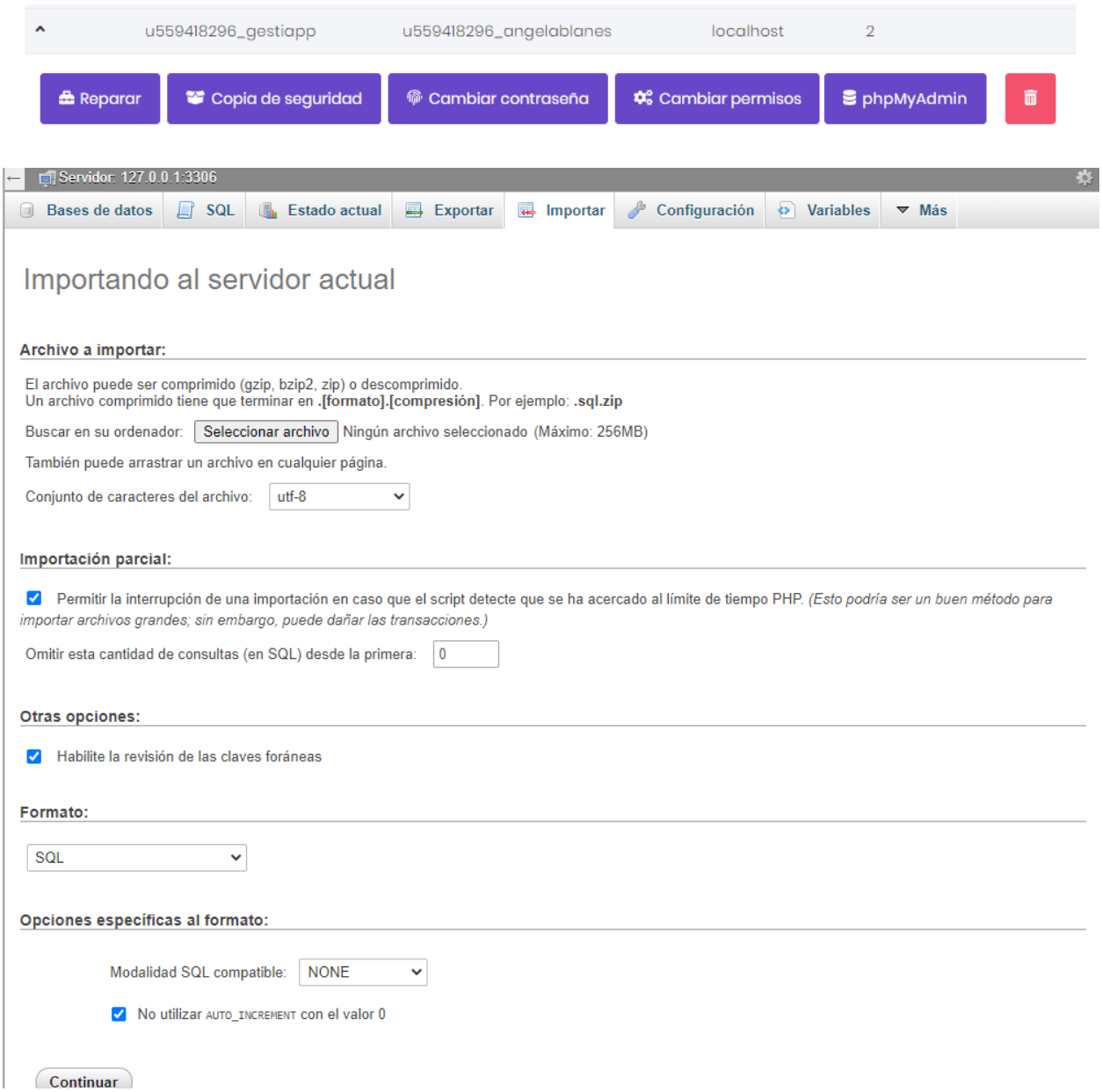

Ilustración 33. Importación base de datos local

Una vez ya tenemos eso, podremos activar el certificado SSL en el dominio, pero deberemos esperar a tener el dominio activado.

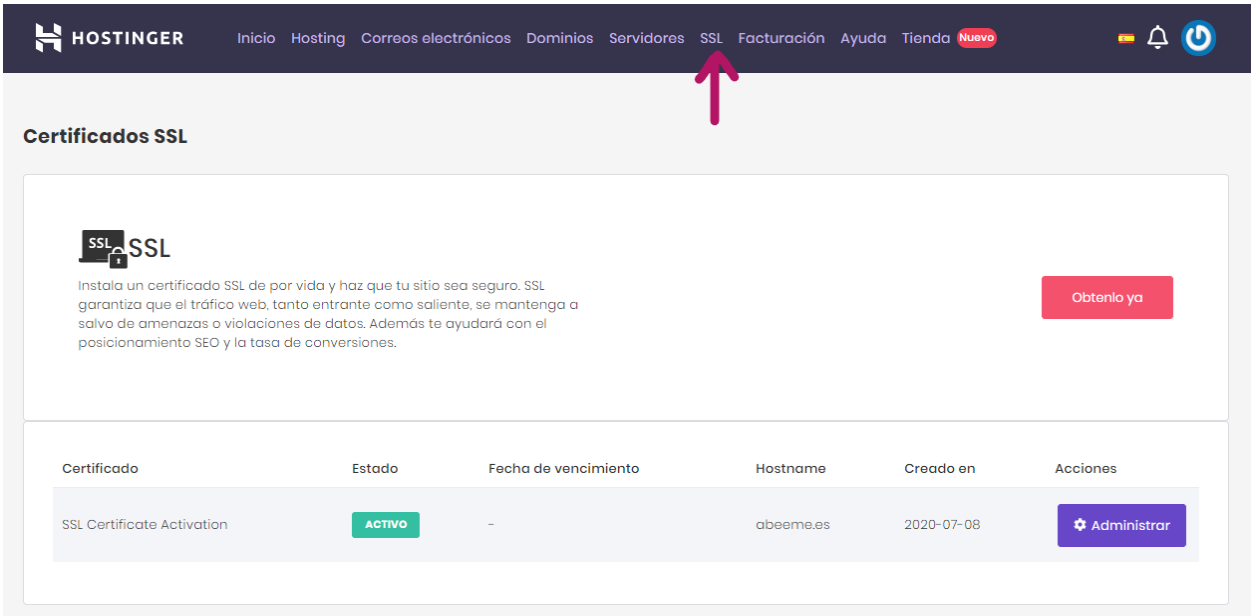

Ilustración 34. Certificado SSL

Y al entrar en Administrar, veremos la instalación en el caso de no estar instalado, y si no, lo que se ve a continuación.

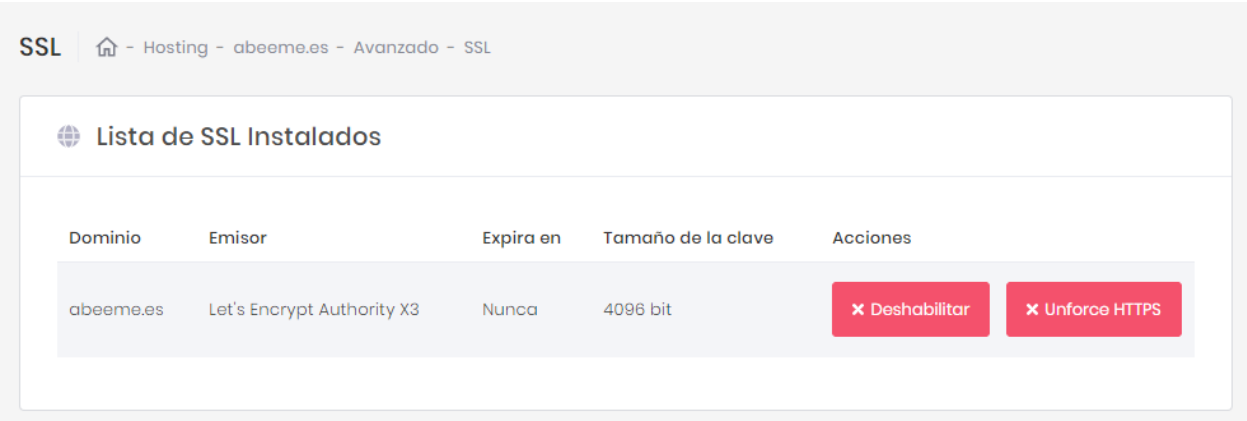

Ilustración 35. Verificación de la instalación del certificado SSL

Cuando ya está todo, dentro del hosting le decimos que queremos crear una nueva página web importando nosotros nuestro proyecto. Y subimos la carpeta con toda la estructura web y nuestros archivos comprimidos en una carpeta zip mediante el administrador de archivos del hosting o un cliente FTP.

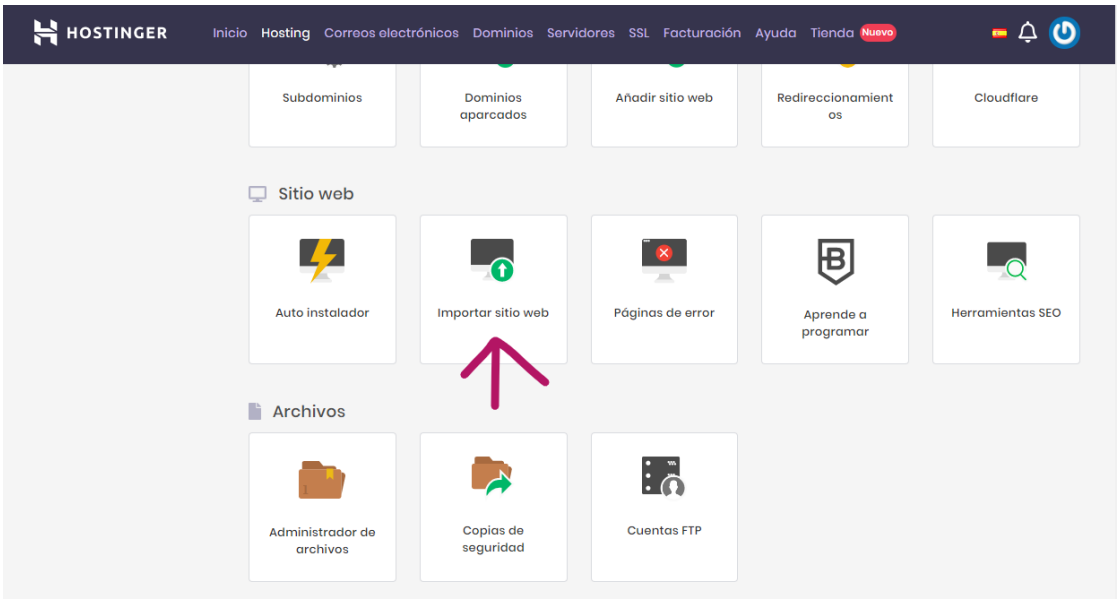

Ilustración 36. Gestor de subida de web

Al tenerla subida, se descomprime en una carpeta y se modifica el fichero de conexion.php que está dentro de scripts en el proyecto y es en el que estará la conexión a la bbdd local, cambiándola por la nueva base de datos creada anteriormente.

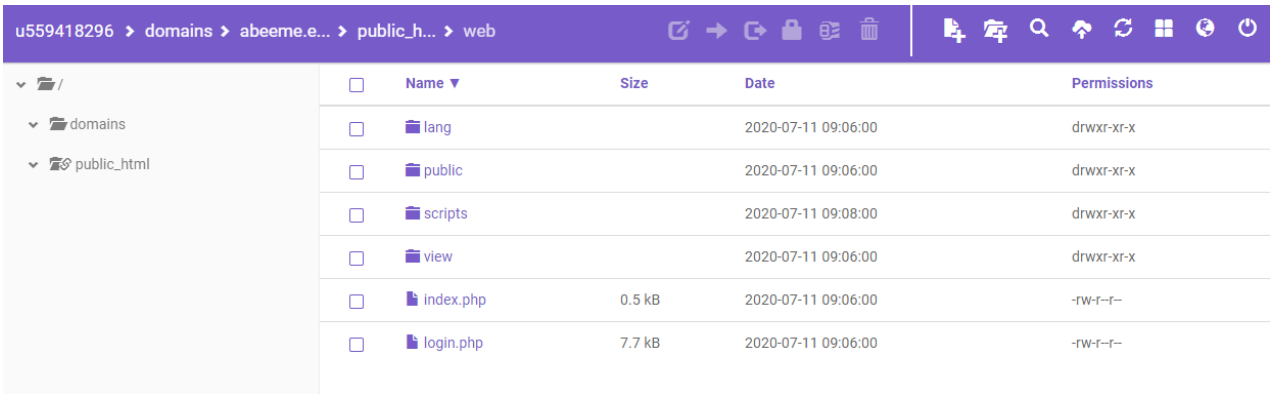

Ilustración 37. Administrador de archivos del hosting

Cuando guardemos, ya podremos entrar en nuestra página web mediante cualquier navegador introduciendo nuestro hosting/la carpeta que hayamos creado.
Aunque en mi caso, fuera de la carpeta, en public, hay un archivo default.php en que le he puesto una redirección, por lo que si en el navegador, se pone solamente el hosting, se redireccionará a la carpeta web, que es donde está el proyecto.

/domains/abeeme.es/public_html/default.php

```php
<?php
    header("Location: /web/");
?>
```

Ilustración 38. Redirección a la web

4.2 Manual del usuario

En este punto, realizaré un manual de usuario utilizando la cuenta de 'jefatura' o 'admin' con la que se pueden ver y realizar todas las acciones, y en los casos en los que el contenido cambie, se especificará para que tipos de usuarios desaparece la opción o la sección.

4.2.1 Instalación de la aplicación

La aplicación web en este caso no tiene instalación previa, ya que como explico en el punto anterior, sería la puesta en producción y una vez instalada ya podría usarse, por lo que no hay pasos previos de instalación.

4.2.2 Explotación

Empezamos y la primera pantalla que nos deberá salir es el Login, donde podremos iniciar sesión con nuestro DNI y una contraseña proporcionadas por algún educador/jefe/administrador.

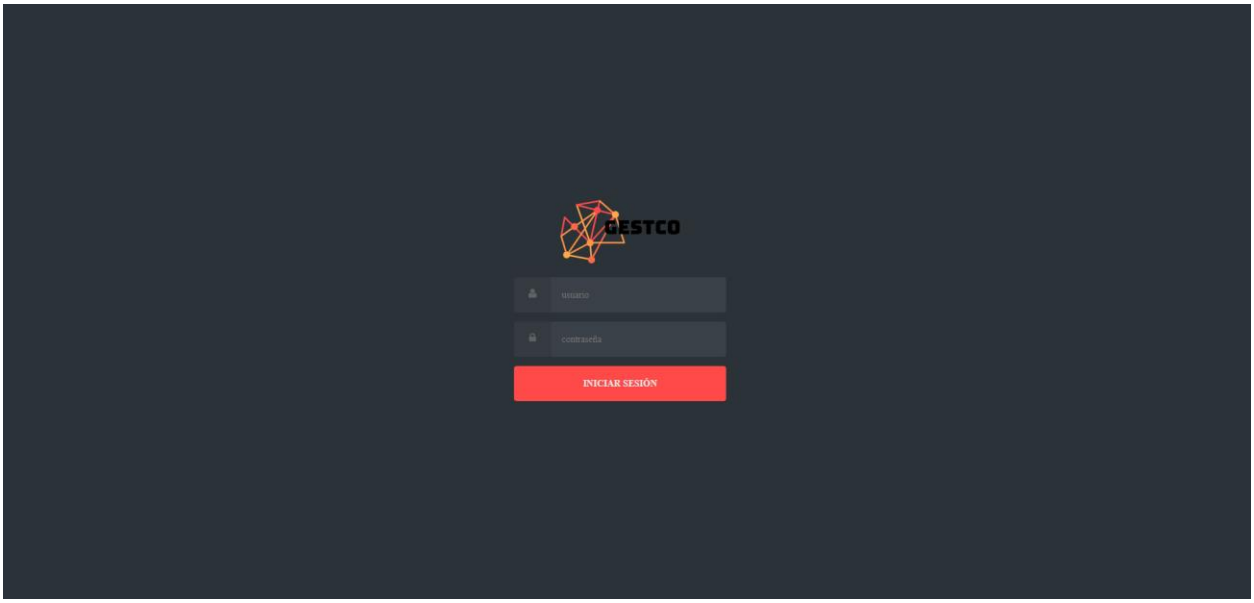

Ilustración 39. Login

Si el usuario existe, nos redireccionará al índice, donde veremos la información personal del usuario que se acaba de loguear, es decir, nuestra información (nombre, apellidos, DNI, correo, enfermedades asociadas al usuario en el caso de que sea o monitor o niño, datos como el correo o teléfonos, los padres asociados y los datos respectivos de cada padre y los hermanos que existan en el centro (solo en los monitores o niños), la información de los hijos (en el caso de ser del tipo padre, madre o tutor)...

Ilustración 40. Página principal

La siguiente ventana será 'Usuarios', y la verán todos los monitores, jefatura y admin.
En ella podremos encontrar una tabla con todos los usuarios dados de alta en la base de datos y que estén activados. Se muestra un resumen con solo algunos de los campos más importantes, pero si nos fijamos, en cada registro, a la derecha, hay unos botones que sirven para:

- El primero, que parece un ojo, nos abrirá un modal con toda la información de este usuario, es decir, igual que en el inicio, nos muestra nuestra información, en este caso mostrará la del usuario.

- El segundo botón será para editar algún campo del usuario, y nos abrirá un modal en el que podremos actualizar la información y volver a guardarlo.

- Y el último, sirve para eliminar el usuario, que en nuestro caso no lo eliminaremos porque queremos tener un historial y un registro, y lo que hará será desactivar el usuario en cuestión.

Además, en la esquina derecha superior, podremos ver otro botón que servirá para dar de alta a un usuario nuevo. Este nos abrirá otro modal en el que nos pedirán los datos necesarios y antes

de guardarlo comprueba si el DNI del usuario ya existiese, y si es el caso de que sí, reactivará el usuario con ese DNI.

Y, por último, vemos que hay un desplegable con todos los sectores que nos servirá para hacer búsquedas por sectores, filtrando y dejándonos ver solamente los usuarios que pertenezcan al sector elegido.

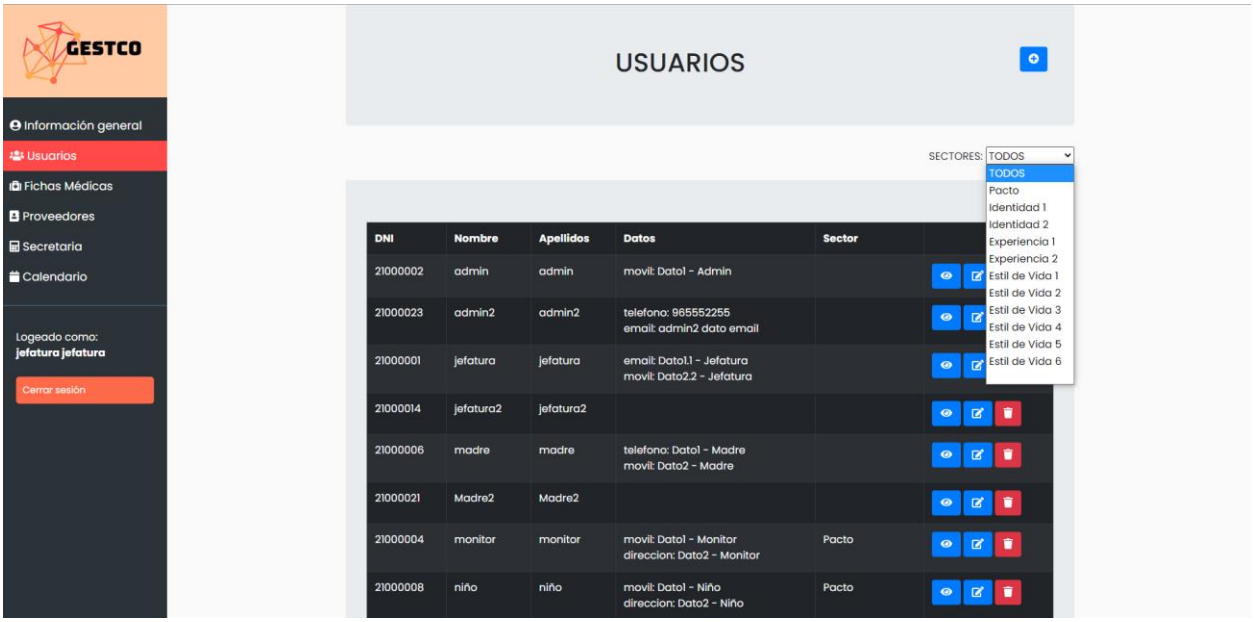

Ilustración 41. Usuarios

Los formularios para dar de alta a un usuario, mostrarán u ocultarán algunos campos dependiendo del tipo de usuario.

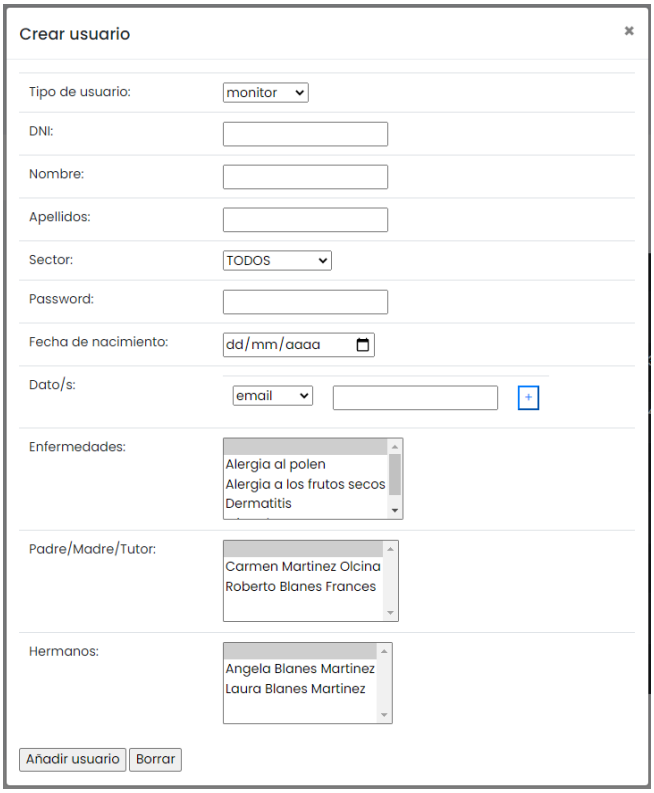

Ilustración 42. Formulario de crear un usuario del tipo monitor

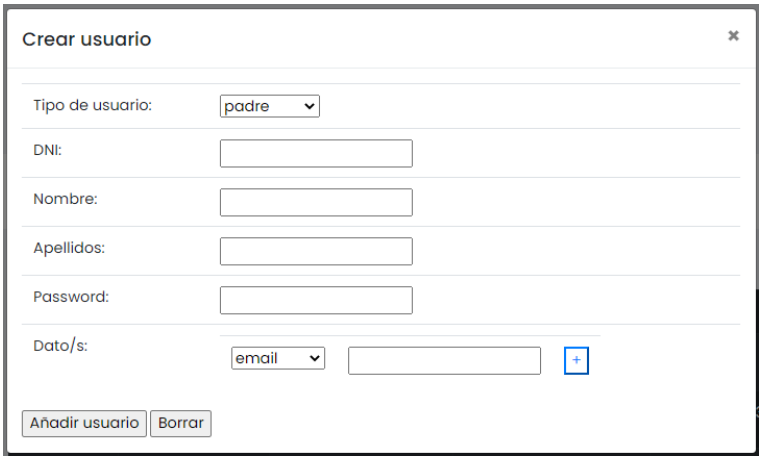

Ilustración 43. Formulario de crear un usuario del tipo padre

Secretaría solo se puede ver por los usuarios de tipo 'secretario', 'jefatura' o 'admin' porque simplemente es una pantalla en la que, a la parte izquierda, podemos ver que pagos actuales hay dados de alta, con la posibilidad de agregar nuevos y seleccionar el que nos interese. Además, también se podrán eliminar.
Y a la parte derecha, hay una lista de todos los usuarios, en la que podemos seleccionar aquellos que hayan pagado el pago seleccionado anteriormente y así asociarlos.

Ilustración 44. Secretaría

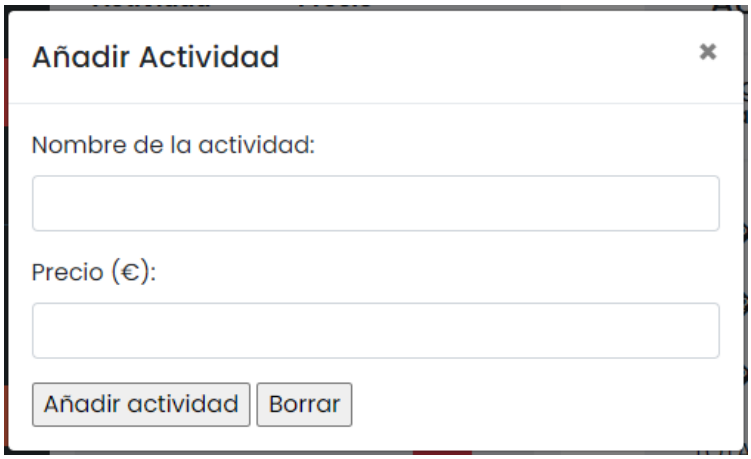

Ilustración 45. Formulario para añadir un pago

En fichas médicas, es un listado de aquellos usuarios con alguna enfermedad o alergia asociada, y que tenemos la opción de volver a filtrar por sector.
Se ha pensado que esta es la mejor manera, ya que, en el caso de necesitar cualquier persona ajena al centro, tener esta información, poder darle de alta el usuario y restringirlo para que solo obtenga este listado.

Ilustración 46. Fichas médicas

En Proveedores, tendrán acceso todos los tipos de usuario menos 'niño', 'padre/madre/tutor' y 'secretaría', y estamos delante de un listado de datos, tanto telefónicos, direcciones o emails, de diferentes empresas de distintos tipo para que nos sea más fácil y lo tengamos todos centralizado, pudiendo filtrar con el desplegable por tipo de empresa (albergues a los que podemos ir, sitios donde se compra comida y bebida al por mayor….).
Además, tenemos la posibilidad de modificar esos datos o dar de alta nuevos proveedores y, lo mismo que en usuarios, comprobará si existe a la hora de añadir uno.

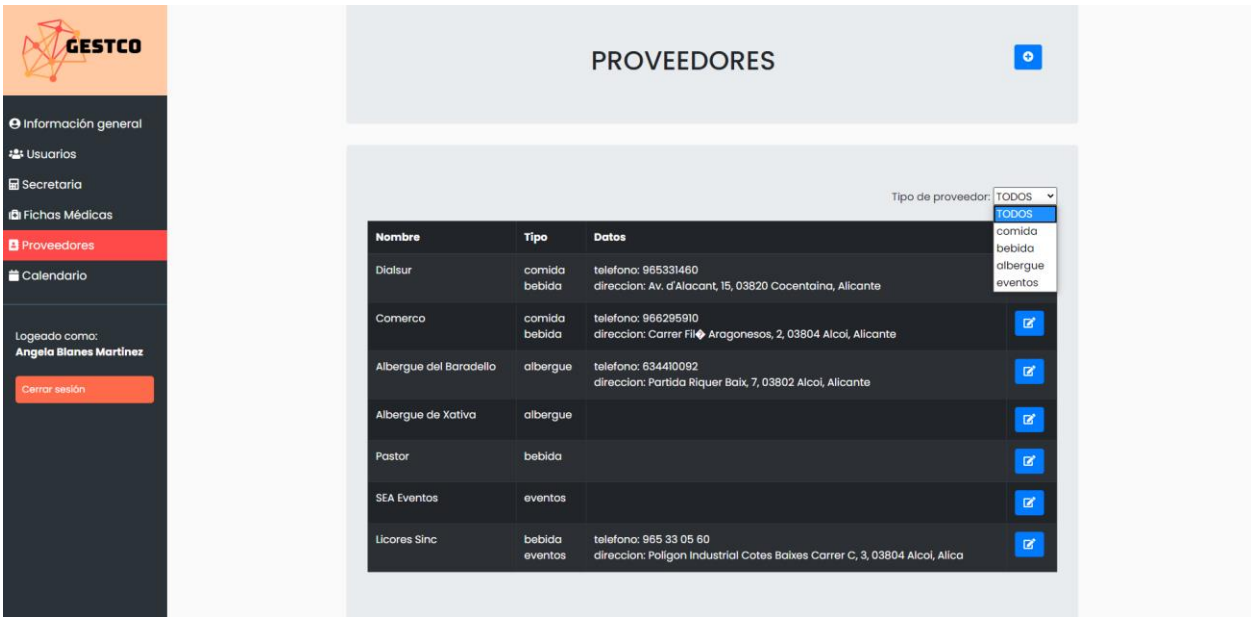

Ilustración 47. Proveedores

Este es el formulario para dar de alta un nuevo proveedor, y sería el mismo para editarlo, lo que se cargan los datos que ya existen.

Ilustración 48. Formulario de añadir un proveedor

Y, por último, tenemos el calendario, en el que se ven todas las actividades del centro, divididas en sectores.
A esta ventana, pueden acceder todos, lo que solo podrán añadir actividades los usuarios con rol 'monitor', 'jefatura' o 'admin'. Y arriba a la derecha, tenemos la información en un modal de qué color tiene cada sector.

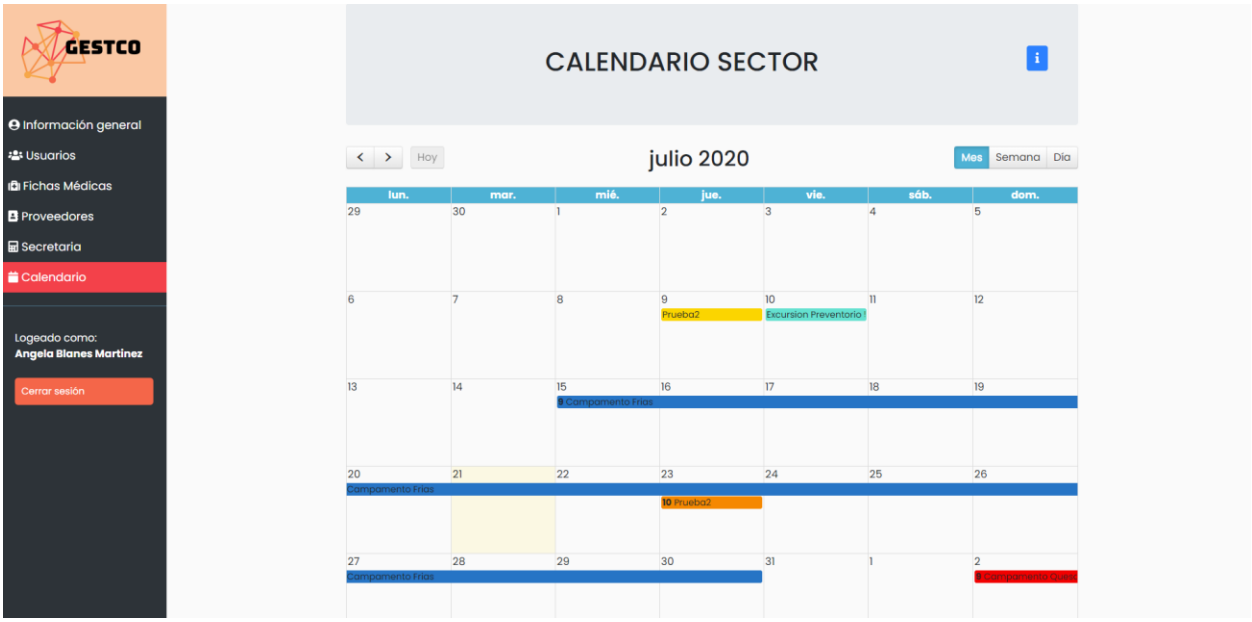

Ilustración 49. Calendario

Para agregar una actividad, tendremos el siguiente modal, que pinchando encima del día que queremos, nos saldrá.

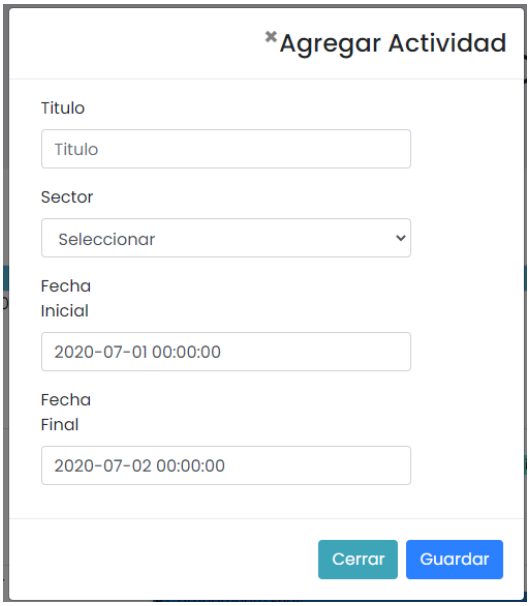

Ilustración 50. Formulario de agregar una nueva actividad

Para editar o eliminar alguna actividad, deberemos hacer doble click encima de ella y editarla. Para eliminarla, se marcará el checkbox y al actualizar, se eliminará del calendario.

Ilustración 51. Formulario de editar o borrar una actividad

Y ya para finalizar, en el menú de navegación, a la parte de abajo, podremos ver el usuario con el que se está logueado y un botón que cierra la sesión en la aplicación.

5 Conclusiones

5.1 Conclusiones personales

Una vez terminada la aplicación web, tengo una sensación de satisfacción por el resultado ya que cumple con todas las expectativas que teníamos el conjunto de monitores y su funcionalidad nos está agilizando mucho el trabajo.

Primero que nada, con las reuniones mantenidas para definir todas las funcionalidades, nos dimos cuenta de muchas carencias que teníamos en el centro y de que ya iba siendo hora de poder tenerlo todo gestionado vía aplicación web e informatizado. Lo que no sabíamos era la envergadura que podría haber tenido este proyecto por no se nos paraban de ocurrir funcionalidades. Pero al final tuvimos también que sentarnos y decidir cuáles eran las más importantes.

Al ser un proyecto que se va a poner en uso, ha sido mucho más fácil imaginarnos los casos que se van a dar e intentar diseñarlo para cumplir con esos posibles flujos. Y, además, eso ha sido una motivación para hacerlo, ya que me siento orgullosa de poder participar y realizar esta idea que se tuvo un día en una cena.

Además, también mirando por mí, es un trabajo en el que he disfrutado y me ha servido, me está sirviendo y me servirá para poder desarrollar más mis conocimientos sobre web y poder aplicarlo en el trabajo. Es un mundo que cada día me apasiona más y en el que creo que nunca dejaré de aprender porque está en constante movimiento y siempre hay cosas nuevas que estudiar.

Y, por último, a la hora de instalar la aplicación en el hosting, he de decir que he aprendido mucho ya no haciéndolo si no informándome de las diferentes alternativas que hay y de los distintos procesos, tipos de hostings, servidores compartidos, servidores dedicados… Porque nunca había tenido la ocasión de instalar una aplicación web de cero en ningún servidor, ya fuera mío o compartido.

En general tengo una sensación muy buena de haber aprendido muchísimo con este proyecto y creo que cara al futuro y a mi futuro, podré aplicar muchos de los conocimientos adquiridos ya que es un mundo muy amplio y que me gusta y esto solo ha servido para corroborar lo que ya creía, y es que la programación web me encanta y no me pasa el tiempo cuando me pongo a ello.

5.2 Futuras líneas de desarrollo

Con la puesta en marcha de esta aplicación, cubre las perspectivas acordadas, pero nos hemos percatado de que se podrían añadir en futuras ampliaciones nuevas funcionalidades, que vamos a mencionar a continuación, que aporten más valor y hagan la aplicación aún más funcional.

- La primera de las nuevas funcionalidades a implementar, podría ser que un usuario que aún no ha sido dado de alta en la plataforma, pudiera hacerlo pinchando en un link, pero como al final, no todo el mundo debería darse de alta, añadiríamos un "filtro" que sería que se notificara al administrador la petición de alta de un usuario y este fuera el que diera la aprobación para la creación final.

- La segunda, sería que se pudiera pagar por la plataforma los diferentes pagos creados y que estos se actualizasen solos en la base de datos al comprobar que se han realizado con éxito.

- El tercero sería la implementación de un módulo que sirviese de inventario en el que registrásemos todo el material del que disponemos (pinturas, pinceles, telas, disfraces…) a partir del escaneo de códigos QR que asignaríamos a cada producto.

- Como cuarta posible futura implementación, sería poder crear una 'Smart app' con la que poder interactuar también con los datos desde un dispositivo móvil.

Aunque mejoras siempre hay, y mejorar una aplicación nunca está de más, conforme la aplicación vaya utilizándose, nos daremos cuenta de las cosas que van haciéndonos falta y siempre quedará abierto el desarrollo a futuros módulos que se necesitasen, en este caso, por ejemplo, uno exclusivo con la información de Campamento...

6 Bibliografía

6.1. Aplicaciones similares

Zoho CRM: https://www.zoho.com/es-xl/crm/ → fecha último acceso (junio, 2020)

SuiteCRM: https://suitecrm.com/ → fecha último acceso (junio, 2020)

SugarCRM: https://www.sugarcrm.com/es/ → fecha último acceso (junio, 2020)

Beyond up: https://bebeyond.es/crm-beyond-up/ → fecha último acceso (junio, 2020)

6.2. Software o librerías utilizadas

Editor de texto → PhpStorm: https://www.jetbrains.com/es-es/phpstorm/ → fecha último acceso (junio, 2020)

Gestor de BBDD → MySQL Workbench: https://www.mysql.com/products/workbench/ → fecha último acceso (junio, 2020)

Servidor web local → Apache: https://www.apachefriends.org/es/index.html → fecha último acceso (junio, 2020)

Diagramas: https://www.gliffy.com/ → fecha último acceso (julio, 2020)

Prototipos: https://proto.io/ → fecha último acceso (julio, 2020)

Hosting: https://www.hostinger.es/tutoriales/ → fecha último acceso (julio, 2020)

Iconos de los botones: https://fontawesome.com/v4.7.0/icons/ → fecha último acceso (julio, 2020)

Control de versiones → GitHub: https://github.com/ → fecha último acceso (junio, 2020)

Calendario JS: https://fullcalendar.io/ → fecha último acceso (julio, 2020)

6.3. Documentación

https://stackoverflow.com/ → fecha último acceso (julio, 2020)

https://www.php.net/ → fecha último acceso (julio, 2020)

https://w3schools.com/ → fecha último acceso (julio, 2020)

https://getbootstrap.com/ → fecha último acceso (julio, 2020)

https://popper.js.org/docs/v2/tutorial/ → fecha último acceso (julio, 2020)

https://css-tricks.com/snippets/css/a-guide-to-flexbox/ → fecha último acceso (julio, 2020)

6.4. Libros

Manual avanzado de desarrollo de aplicaciones Web, Moseley, Ralph | Madrid : Anaya Multimedia, D.L. 2007.

7 Acrónimos

- API: (Interfaz de programación de aplicaciones) Conjunto de subrutinas, funciones y procedimientos que se incluyen en las bibliotecas para ser utilizados como capa de abstracción en otros programas.

- HTTP (HyperText Transfer Protocol): Protocolo utilizado en cada transacción de la web.

- HTTPS: Es la versión segura de HTTP

- IDE (Integrated Development Environment): Aplicación informática facilita el desarrollo del software al programador.

- JS: diminutivo para referirse a JavaScript

- SQL: Lenguaje declarativo de acceso a bases de datos relacionales con el que se puede especificar varios tipos de operaciones.

- SSL (Secure Sockets Layer): tecnología estándar para mantener segura una conexión a internet y para proteger la información confidencial que se envía entre dos sistemas.

- CMS (Content Management System): sistema de gestión de contenidos desarrollado para que cualquier usuario pueda administrar y gestionar contenidos de una web fácilmente y sin saber programar.