Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

CONTROL DE ASISTENCIA

Alumno: Antonio Díaz Herrera

Tutor: Prof. D. JUAN ROBERTO JIMENEZ PÉREZ

Dpto: Informática

SEPTIEMBRE,2019

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don JUAN ROBERTO JIMENEZ PÉREZ, tutor del Proyecto Fin de Carrera titulado: CONTROL DE ASISTENCIA, que presenta ANTONIO DIAZ HERRERA, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2019

El alumno: Los tutores:

ANTONIO DIAZ HERRERA JUAN ROBERTO JIMENEZ PÉREZ

Índice

Índice de Ilustraciones

Índice de Tablas

RESUMEN

El propósito de este proyecto es el de crear un prototipo en el que se puedan guardar y consultar las horas de trabajo. Para ello habrá que crear una parte hardware capaz de identificar las horas a las que unos usuarios pasan unos identificadores por él, y una parte software, en la que los usuarios tendrán un perfil propio y podrán consultar el número de horas trabajadas a lo largo del tiempo, y según el nivel de acceso de dicho usuario, las opciones de dar de alta o baja a usuarios o crear nuevos grupos de trabajo.

1. Capítulo 1. Introducción

1.1. Motivación

A día de hoy, todas las empresas necesitan llevar un control sobre sus trabajadores como se estipula en el apartado V del Real Decreto-ley 8/2019, https://www.boe.es/buscar/doc.php?id=BOE-A-2019-3481, en el cual a partir del día 8 de marzo de 2019 todas las empresas están obligadas a llevar un registro de las horas de trabajo de sus empleados, independientemente de la jornada que estos tengan. Es por ello que he decidido crear un sistema capaz de almacenar toda la información relacionada, tanto de trabajadores, con información sobre el nombre, apellidos, DNI, etc, como de las horas de trabajo, mes, día, hora. El propósito de este proyecto es que tanto los trabajadores, como sus responsables, puedan llevar un control sobre las horas de trabajo en cualquier momento y en cualquier lugar. En la mayoría de estos sistemas, no se informa sobre el número de horas trabajadas hasta que no se finaliza cada mes.

1.2. Objetivos

Los objetivos que se esperan conseguir con la realización de este proyecto son:

- Seleccionar tecnologías: Recabar información sobre las distintas tecnologías de manera que sea capaz de seleccionar aquellas que se adecuen lo mejor posible a nuestras necesidades.
- Desarrollo de una página web: Ser capaces de hacer una página web con distintas vistas según el tipo de usuario iniciado y en la que se muestre toda la información necesaria para el proyecto.
- Crear una Base de Datos: Construir una base de datos consistente donde se almacene toda la información sobre los distintos usuarios, los distintos departamentos de dicha empresa y las horas en las que se realizan los fichajes.
- Construir un sistema Hardware de identificación: Crear un dispositivo, el cual sea capaz de enviar información al servidor y almacenarlo en la base de datos, a la vez que interactúa con el usuario.
- Redactar una memoria: Redactar una memoria sobre el proyecto y todas sus fases.

1.3. Resultados esperados

Tras la realización del proyecto se espera poder tener un sistema hardware usable capaz de almacenar las entradas y salidas de los diferentes usuarios a través de dispositivos NFC, así como una página web en la que los usuarios puedan identificarse y visualizar, según su grado de accesibilidad, información asociada.

1.4. Estructura del Proyecto

A continuación, se indica la organización de la memoria y se proporciona una breve descripción de cada apartado::

1. Introducción: En este apartado se cuentan las razones que me han motivado a elegir este trabajo, así como los objetivos que se esperan llevar a la finalización del mismo y un desglose de la estructura del proyecto.
2. Metodologías, Planificación y Costes: Este capítulo muestra una comparación de las diferentes metodologías disponibles, así como una

planificación del proyecto y un apartado donde se muestran los costes tanto hardware, software y coste de personal para llevar a cabo la realización de este proyecto.

3. Tecnologías y Componentes: Aquí se hace una mención a las distintas tecnologías disponibles, un estudio y razones por las que finalmente he seleccionado para llevar a cabo el proyecto tanto para el módulo de identificación, como para la página web y el uso de software complementario para la realización de la misma.
4. Desarrollo: A partir de este capítulo se comienza a explicar la implementación de las distintas fases del modelo seleccionado en el capítulo anterior, es el capítulo más extenso de la memoria ya que a su vez es el que más se ha prolongado en el tiempo.
5. Conclusiones y mejoras: En este capítulo se expondrán las impresiones obtenidas a una vez acabado el trabajo y un conjunto de posibles mejoras del proyecto para realizar en el futuro.
6. Bibliografía: Se hará un desglose se los diferentes referencias que he utilizado a la hora de realizar el trabajo.
7. Anexos: Llegamos a la parte final de la memoria que constara de tres partes. La primera estará formada por el manual de instalación de la aplicación. La segunda parte mostrará un manual de usuario y por último en la tercera se comentará el contenido que se adjuntará en el CD del proyecto.

2. Capítulo 2. Metodologías, Planificación y Costes

2.1. Metodologías

"Son muchos los modelos que se han descrito en la literatura sobre ingeniería del software. Algunos son prescripciones para la manera en que debe avanzar el desarrollo del software, y otras son descripciones de la manera en que le desarrollo del software se hace en la realidad. En construcción de un modelo de proceso y la discusión de los subprocesos ayuda al equipo a comprender la brecha que existe entre lo que debe ser y lo que es." (Pfleeger, 2002)

Existen otras razones para el modelado de un proceso:

• Cuando un grupo pone por escrito una descripción de su proceso de desarrollo, da forma a una comprensión común de la actividades, recursos y restricciones comprometidos en el desarrollo del software.

• La creación de un modelo de proceso ayuda al equipo de desarrollo a encontrar las inconsistencias, las redundancias y las omisiones en el proceso y en las partes que lo constituyen. A medida que estos problemas se descubren y se corrigen, el proceso se vuelve más efectivo y concentrado en la construcción del producto final.

• El modelo debe reflejar las metas del desarrollo, como la construcción de software de alta calidad, la localización temprana de los defectos en el desarrollo y el cumplimiento de las restricciones del cronograma y el presupuesto. A medida que se construye el modelo. El equipo de desarrollo evalúa las actividades candidatas por su grado de adecuación para alcanzar estas metas. Por ejemplo, el equipo puede llevar a cabo revisiones de requerimientos, de modo que los problemas relativos al requerimiento puedan encontrarse y subsanarse antes de dar comienzo al diseño.

• Todo proceso debe ser adaptado para la situación especial en que será utilizado. La construcción de un modelo de proceso ayuda al equipo de desarrollo a comprender dónde debe hacerse la adaptación.

• Todo modelo de proceso de desarrollo de software incluye los requerimientos del sistema como entrada y un producto entregado como salida.

2.1.1. Modelo en Cascada

Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase (Ilustración 1). Sin embargo, este modelo presenta problemas frecuentes que afectarían al desarrollo de nuestro proyecto, entre los que se encuentran los siguientes:

1. Los proyectos reales no suelen seguir el flujo secuencial propuesto por el modelo. Como resultado los cambios provocan confusión conforme el equipo del proyecto avanza.
2. Suele ser difícil para el cliente enunciar todos los requerimientos de forma explícita al comienzo del proyecto, y al ser necesario en el modelo en cascada, presenta dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos.
3. El cliente debe tener paciencia, no se dispondrá de una versión funcional del programa hasta que el proyecto esté muy avanzado.

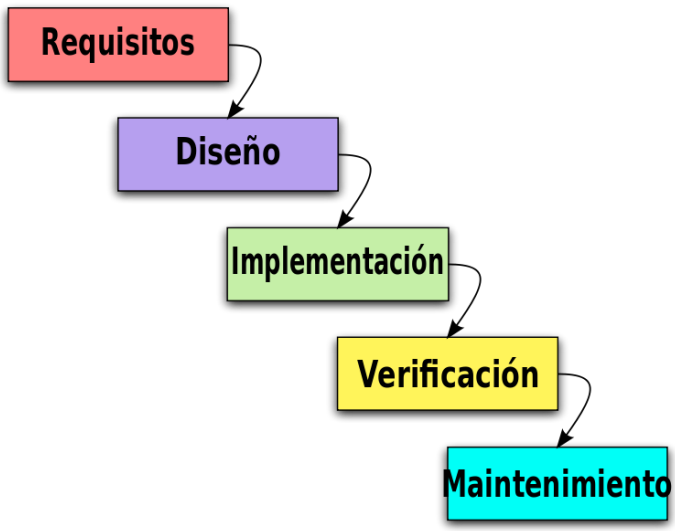


Ilustración 1. Modelo en Cascada

2.1.2. Modelo de procesos evolutivo.

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software. En este apartado vamos a ver dos de los modelos más comunes de proceso evolutivo.

2.1.3. Hacer Prototipos

Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente (Ilustración 2).

Inconvenientes:

1. El usuario tiende a crearse unas expectativas cuando ve el prototipo de cara al sistema final. A causa de la intención de crear un prototipo de forma rápida, se suelen desatender aspectos importantes, tales como la calidad y el mantenimiento a largo plazo, lo que obliga en la mayor parte de los casos a reconstruirlo una vez que el prototipo ha cumplido su función. Es frecuente que el usuario se muestre reacio a ello y pida que sobre ese prototipo se construya el sistema final, lo que lo convertiría en un prototipo evolutivo, pero partiendo de un estado poco recomendado.
2. El desarrollador suele tomar algunas decisiones de implementación poco convenientes. Con el paso del tiempo, el desarrollador puede olvidarse de la razón que le llevó a tomar tales decisiones, con lo que se corre el riesgo de que dichas elecciones pasen a formar parte del sistema final.

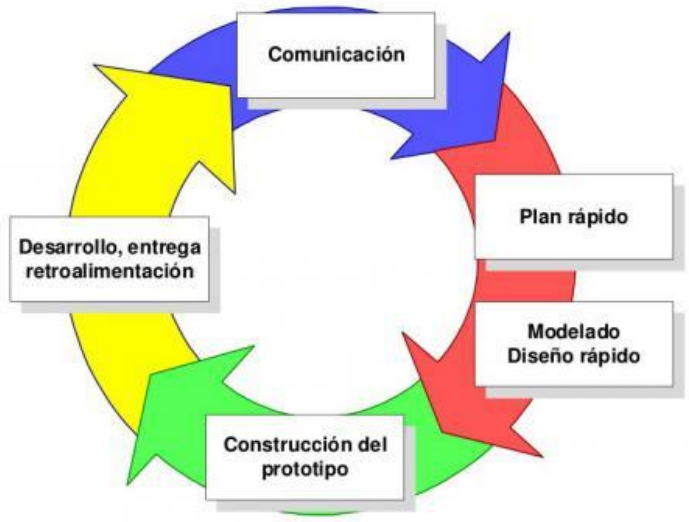


Ilustración 2. Modelo evolutivo: Hacer prototipos

2.1.4. Modelo en espiral

El modelo en espiral (Ilustración 3) se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. El software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completes del sistema cuya ingeniería se está haciendo. En la Ilustración 3 puede verse un esquema de este modelo.

Inconvenientes:

3. Debido a su elevada complejidad no se aconseja utilizarlo en pequeños sistemas.
4. A menudo sucede que los prototipos se transfieren al sistema de producción. Por lo tanto, existe el riesgo de que se introduzcan otros errores e incoherencias conceptuales en el producto final posterior
5. En los lugares donde se toman decisiones sobre los ciclos siguientes, existe el riesgo de que se formen bucles y el proyecto tarde más tiempo si se toman decisiones equivocadas.

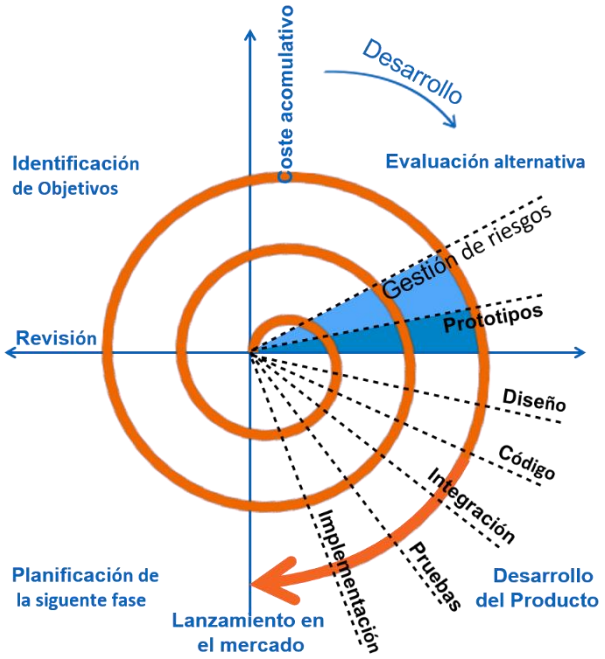


Ilustración 3. Modelo en espiral

2.1.5. Modelo Incremental

Cuando se utiliza el modelo incremental (Ilustración 4Ilustración 4. Modelo Incremental), el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

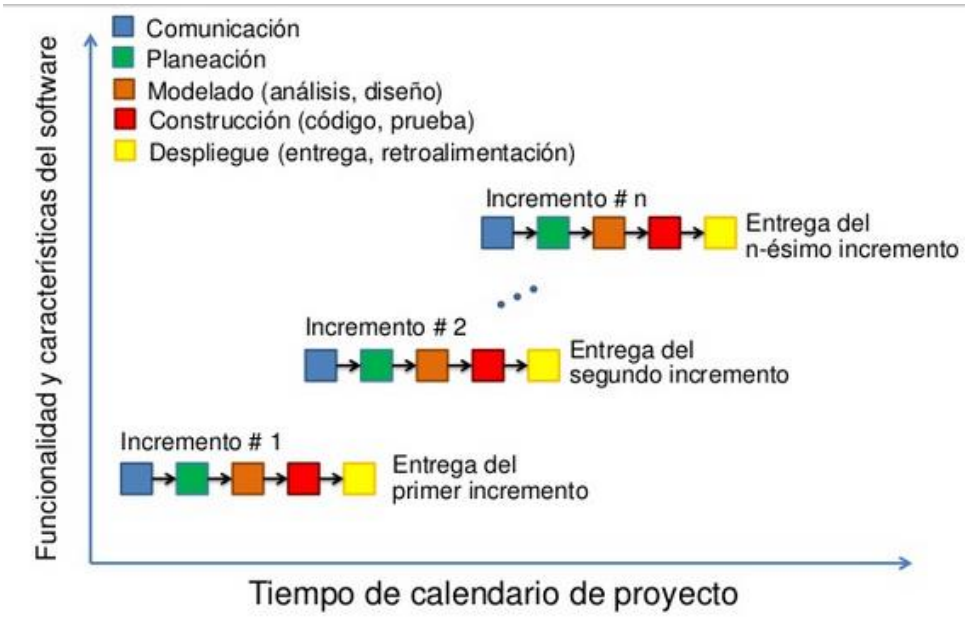


Ilustración 4. Modelo Incremental

2.1.6. Selección de un Modelo

Examinando los distintos modelos existentes he llegado a la conclusión de que el modelo incremental es el que mejor se adecua para la realización del proyecto debido a que combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Uno de los puntos fuertes de este modelo y que me ayudo a decantarme por él, es que desde el inicio se obtiene un prototipo funcional a partir del cual se van añadiendo más funcionalidades. Esto permite observar fallos en el diseño o nos ayuda a añadir funcionalidades que a priori no nos parecían necesarias o ni siquiera se nos habían ocurrido para implementarlas en nuestro producto.

Entre las ventajas que puede proporcionar un modelo de este tipo encontramos las siguientes:

1. Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software, por lo que el usuario se involucra más.
2. Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.
3. Es más fácil probar y depurar en una iteración más pequeña.
4. Es más fácil gestionar riesgos, tanto los relacionados con el tiempo como errores en el diseño, ya que al funcionar por incrementos, los errores cometidos en incrementos anteriores se puede solucionar en incrementos futuros.
5. Cada iteración es un hito gestionado fácilmente.

2.2. Planificación temporal

En este apartado se verán aspectos relacionados con la cuantificación del tiempo que se invertirá en la creación del proyecto.

2.2.1. Diagrama de Gantt

"El diagrama de Gantt es una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado" (Wikipedia). La duración de dichas tareas se realizará en días y a partir de la Tabla 1 obtendremos el citado diagrama, el cual se puede ver en la Ilustración 5.

Tarea	Duración	Inicio	Final
Inicio	1 día	12/03/2018	12/03/2018
Planificación	3 días	13/03/2018	15/03/2018
Incremento 1	6 días	16/03/2018	21/03/2018
Incremento 2	8 días	22/03/2018	02/04/2018
Incremento 3	10 días	03/04/2018	16/04/2018
Incremento 4	12 días	17/04/2018	02/05/2018
Incremento 5	4 días	03/05/2018	08/05/2018
Pruebas	5 días	09/05/2018	15/05/2018
Redactar Memoria	55 días	12/03/2018	25/05/2018

Tabla 1. Tabla diagrama de Gantt

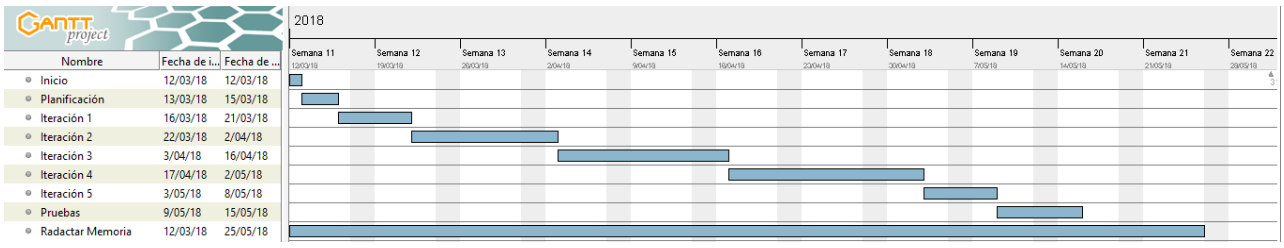


Ilustración 5. Diagrama de Gantt.

2.3. Costes

El cálculo de los costes nos va a permitir conocer por adelantado los gastos de nuestro proyecto, desde la planificación inicial hasta su entrega. Para obtener el coste total del proyecto, se realizará una suma de los costes relacionados con el hardware, software, coste del personal y otros costes no relacionados con estos campos, como pueden ser las herramientas para crear el sistema de identificación.

2.3.1. Costes Hardware

Es el coste relacionado con los componentes necesarios para crear el sistema de identificación, así como los costes hardware a partir de los cuales crearé la página web. En la Tabla 2 se muestran los componentes utilizados y el coste de los mismos.

Componentes Hardware: Sistema de Identificación	**Coste**
Módulo Pantalla LCD	7.99€
Pcf8574 i/o	10.00€
W5100 Ethernet Shield NodeMcu	6.90€
PN532 NFC	12.69€
NodeMCU Lua Lolin V3 ESP8266	7.69€
Cable ethernet	5.99€
PCB	11.99€
VANYE cable dupont	6.99€
TOTAL	**70.24€.**

Tabla 2. Tabla Hardware Sistema de Fichaje

Como hardware para la realización de la página web he utilizado un ordenador Toshiba, modelo SATELLITE C55-A-1NV con un Procesador Intel Core i5-4200M, 8 Gb de RAM, su precio de compra fue de 899€. El equipo no ha sido adquirido expresamente para el proyecto, por lo que hay que calcular la parte proporcional. La vida útil del dicho equipo suele ser 5 años por lo que el coste por día es de 0.4926€. Como la duración del proyecto es de 55 días es coste final será de 55x0.4926=27.093€. También será necesaria una conexión a internet, la cual tiene un coste de 70€ al mes, por lo que al día costaría unos 2.33€. Por lo tanto, el precio final será de 55 x 2.33€ =128€. En la Tabla 3 se muestra la suma de todos estos costes.

Componentes Hardware Final	**Coste**
Ordenador	27.093€
Conexión a internet	128€
Hardware de Identificación	70.24€
TOTAL	**167.093€**

Tabla 3. Tabla Hardware Página web

2.3.2. Costes Software

En la Tabla 4 aparecen todas las aplicaciones utilizadas a lo largo del proyecto tanto para desarrollo de código, como para la creación del sistema, así como su coste.

Desarrollo	**Software**	**Coste**
Base de Datos	PhpMyAdmin	0.00€
Framework	CodeIgniter 3.0	0.00€
Virtualización	Xampp	0.00€
Modelo 3D	3D Builder	0.00€

Impresión Modelo 3D	Kura	0.00€
Diseño	Visual Paradigm	19.00€
Microsoft Office 19	Microsoft Office 19	7 €/mes x 2 meses= 14.00€
Planificación	GanttProject	0.00€
	TOTAL	**33.00€**

Tabla 4. Tabla Coste Software

2.3.3. Costes del Personal

Se necesitarán un ingeniero de software y un programador junior para la realización del proyecto. El salario de un ingeniero de software suele variar bastante según la información que he podido encontrar, por lo tanto, hare una estimación media según los datos que he encontrado y en nuestro caso, el precio por hora de un ingeniero de software nos será de 15€/hora. Dado que de los 55 días que dura el proyecto se han necesitado 11 días para analizar y diseñar el sistema, el precio final del ingeniero de software será de 15€/hora x 8 horas x 11 días es igual a 1.320€. Por otra parte, el coste por hora de un programador Junior suele estar sobre los 8€/hora, que ha necesitado 44 días para realizar el proyecto y que ha trabajado 8 horas al día, esto hace un total de 352 horas. Por lo que el coste del programador será de 352x8€= 2.816€. Por último, hay que sumar ambas partes para obtener el coste final del personal, que sería de 1.320€+2.816€=4136€.

2.3.4. Otros Costes

A este apartado pertenecen aquellos costes relacionados con la creación de la carcasa donde se depositará el sistema de fichaje. La carcasa se creará a parir de una impresora 3D por lo que su precio será una aproximación en cuanto al gasto de filamento necesario y uso de la impresora. (Tabla 5)

Componente	Precio
Tornillos y tuercas m3	0.99€
Cinta adhesiva doble cara	6.95€

Herramientas	21.24€
Impresión de carcasa	10.00€
Estaño	4.10€
TOTAL	**43.28€**

Tabla 5. Tabla de Otros Costes

El resultado final del proyecto será el mostrado en la Tabla 6.

Coste	**Precio**
Hardware	167.093€
Software	33.00€
Personal	4136.00€
Otros	43.28€
TOTAL	**4379.373€**

Tabla 6. Coste Final.

3. Capítulo 3. Tecnologías y Componentes.

En este apartado se explicarán las diferentes tecnologías seleccionadas para la realización de proyecto, así como de los componentes hardware para la realización del módulo de identificación. Es por eso que he decidido dividir este capítulo en dos partes: en la primera se tratara todo lo relacionado con los componentes tanto hardware como software para crear el módulo de identificación y en la segunda todo lo relacionado con el software para la realización de la página web.

3.1. Módulo de Identificación

3.1.1. NodeMCU esp8266

NodeMCU es una placa de desarrollo totalmente abierta, a nivel de software y de hardware. El NodeMCU no es un microcontrolador al igual que Arduino, son placas o kits de desarrollo que llevan incorporados un chip que se suele llamar SoC (Sytem on a Chip) que dentro tiene un microcontrolador o MCU. (Ilustración 6)

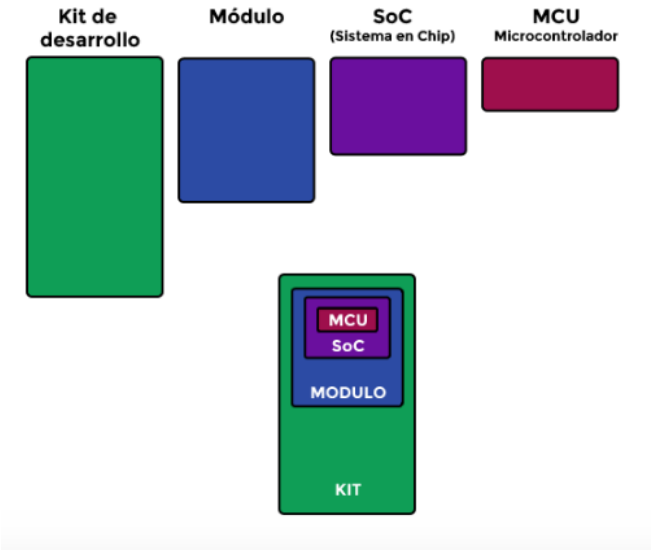


Ilustración 6. Esquema placas NodeMcu

La mayor ventaja que tienen placas es que incorporan un módulo WiFi que nos permite crear proyectos del IoT o sistemas inalámbricos.

En la Ilustración 7 puede observarse como es físicamente el NodeMCU esp8266 cuyas características principales son:

- Conversor Serie-USB para poder programar y alimentar a través del USB.
- Fácil acceso a los pines.
- Pines de alimentación para sensores y componentes.
- LEDs para indicar estado.
- Botón de reset.
- Velocidad 80 MHZ.
- Incorpora una MCU de 32-bit de bajo consumo.
- Módulo WiFi de 2.4 GHz.
- RAM de unos 50 kB.
- 1 entrada analógica de 10-bit.
- 17 pines de entrada y salida GPIO (de propósito general).

Ilustración 7. NodeMCU esp8266

Pines

Para la conexión con otros componentes disponemos de los siguientes pines (Ilustración 8):

- 13 pines digitales numerados del D0 al D12
- 1 pin analógico numerado A0
- 3 pines de 3,3V

- 1 pin de 5V (versión V3 2 pines 5V)
- 4 pines de tierra GND (versión V3 5 pines GND)

Ilustración 8. Pines NodeMcu.

Además, existen pines de alimentación tienen las siguientes funciones:

- Alimentar sensores y componentes (salida)
- Alimentar la propia placa (entrada)

El voltaje de operación de NodeMCU es de 3,3V, por lo tanto, en principio no podríamos alimentar ningún componente que necesitara 5V.

Sin embargo, cuando se alimenta a través del puerto USB con 5V, internamente tiene un regulador de voltaje que saca 3,3V y 5V. Los 3,3V se utilizan para alimentar el NodeMCU y para sacarlo por los 3 pines marcados con ese valor.

Los 5V se utilizan para alimentar otros componentes dentro de la placa y para sacarlos por el pin de 5V.

Pulsadores

Además de los LEDs la placa tiene dos pulsadores. El típico botón de RESET (RST) que si lo pulsamos se resetea la placa y comienza la ejecución de cero y otro botón que pone FLASH, los cuales pueden apreciarse en la Ilustración 9Ilustración 8.

Ilustración 9. Pulsadores NodeMcu.

El botón de RESET hace eso, resetear. Esto no quiere decir que elimine el código.

El botón de FLASH nos permite cargar un programa o firmware. Esto no es algo específico de NodeMCU o del ESP8266, todos los microcontroladores tienen como mínimo dos estados.

3.1.2. Tecnología NFC

Es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos. Los estándares de NFC cubren protocolos de comunicación y formatos de intercambio de datos.

Esta tecnología se comunica mediante inducción en un campo magnético, en el que dos antenas de espiral son colocadas dentro de sus respectivos campos cercanos. Trabaja en la banda de los 13,56 MHz, esto hace que no se aplique ninguna restricción y no requiera ninguna licencia para su uso. En la Ilustración 10 pueden observarse u ejemplo de tarjetas que utilizan esta tecnología.

Ilustración 10. Tarjetas NFC.

Soporta dos modos de funcionamiento:

- Activo: ambos dispositivos generan su propio campo electromagnético, que utilizarán para transmitir sus datos.

- Pasivo: solo un dispositivo genera el campo electromagnético y el otro se aprovecha de la modulación de la carga para poder transferir los datos. El iniciador de la comunicación es el encargado de generar el campo electromagnético.

3.1.3. NodeMcu IDE

Para la creación del código del módulo de identificación, he utilizado el software Arduino IDE v1.8.5 (Ilustración 11). Es un software libre y fácil de usar, que además

nos permite utilizar librerías internas, como importar librerías creadas por otros usuarios. Solo se necesita conectar la placa NodeMcu a través de un USB al ordenador y escoger en el programa el tipo de NodeMcu y la frecuencia de subida para empezar a trabajar con él.

Ilustración 11. Logo Arduino IDE

3.1.4. 3D Builder

Para la creación de la carcasa he utilizado dos programas distintos, uno para crear el modelo y otro para la impresión del mismo. El primer software utilizado viene ya instalado en nuestro equipo. 3DBuilder (Ilustración 12) es una herramienta perfecta para crear modelos 3D ya sea creando el modelo desde cero o haciendo modificación en otros previamente importados. Es una herramienta fácil de utilizar y nos permite exportar el modelo en diferentes formatos, incluyendo el ACI, necesario para poder utilizarlo posteriormente en el software de impresión.

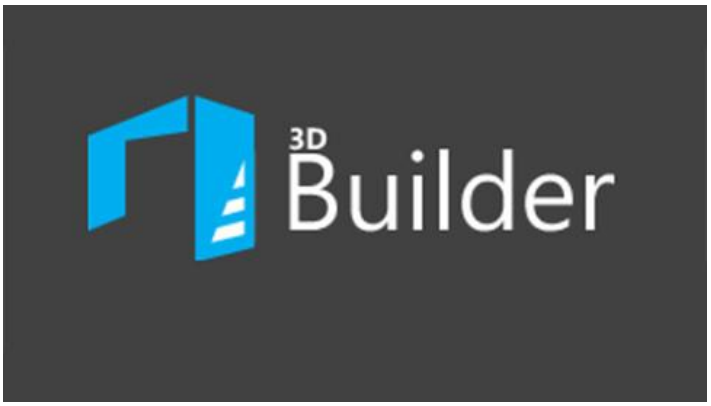


Ilustración 12. Logo 3DBuilder

Para el software de impresión he utilizado la herramienta Ultimaker Cura 3.0 (Ilustración 13), es una herramienta para impresión 3d fácil de utilizar pero que a su vez nos permite modificar las propiedades de impresión para seleccionar aquellas que mejor se adecuen a nuestras necesidades, como por ejemplo la velocidad de impresión, la cantidad de grosor en la impresión, el número de capas, densidad de relleno, temperatura de impresión, adherencia a la placa entre otras.

Ilustración 13. Logo Cura

3.2. Página Web

3.2.1. CodeIgniter

Me he decantado por utilizar el framework CodeIgniter, ya que tengo experiencia de trabajo con él, se basa libremente en el popular patrón de desarrollo model-view-controller (MVC) el cual es idóneo para los resultados que queremos obtener con nuestra página, además de que dispone de diferentes librerías para garantizar seguridad y poder construir una API para así comunicarse con el sistema de fichaje.

3.2.2. Boostrap

Es un framework CSS gratuito y open-source, responsivo, mobile-first para desarrollo web front-end. Contiene CSS y plantillas basadas en JavaScript para tipografías, botones, formularios, navegación y otros componentes de interfaz.

En este proyecto, se ha utilizado la versión 3 de Bootstrap, en concreto he utilizado el siguiente estilo https://themes.getbootstrap.com/product/beagle-responsive-admin-template/ para la creación de los formularios, tablas, y el resto de

elementos visuales que se encuentran en la aplicación, tales como los modales o las alertas.

Es un lenguaje de hoja de estilo usado para describir la presentación de un documento escrito en un lenguaje de marcado como HTML (Ilustración 14).

Ilustración 14. Logo Bootstrap.

3.2.3. CSS

CSS se ha diseñado para permitir la separación entre presentación y contenido, incluyendo layout, colores y fuentes. Esta separación mejora la accesibilidad del contenido, proporciona más flexibilidad y control en la especificación de las características de presentación, permite que múltiples páginas web compartan formato, especificando el CSS relevante en un fichero .css separado, y reduce la complejidad y la repetición en el contenido estructural.

Se ha utilizado junto al código HTML para proporcionar a las vistas de estilo.

3.2.3. PHP

PHP (Ilustración 15) ha sido el lenguaje de programación utilizado por excelencia para el proyecto, en su versión 7.1. Es un lenguaje de código abierto adecuado para el desarrollo de aplicaciones web dinámicas, y puede ser incrustado en HTML.

PHP se ejecuta en el lado del servidor, por lo que genera el HTML y se lo envía al cliente, por tanto el cliente recibe el resultado del script. El código es interpretado

por un servidor web que genera el texto plano en formato UTF-8. Dicho código PHP es invisible al navegador web y al cliente.

PHP está pensado para emplearse en todos los SSOO. También admite la mayoría de servidores web que existen.

Ilustración 15. Logo PHP.

3.2.4. HTML

HTML5 se ha utilizado para la creación de las vistas de la aplicación (Ilustración 16).

Es el lenguaje de marcado para la elaboración de páginas web. Se puede complementar con tecnologías como CSS y lenguajes de scripts como JavaScript.

HTML5 es la quinta y actual versión de HTML. Incluye modelos de procesamiento detallados, para desarrollar implementaciones más interoperables, extiende, mejora y racionaliza el marcado disponible para los documentos, e introduce marcado e interfaces de programación de aplicaciones (APIs) para las aplicaciones web más complejas.

Ilustración 16. Logo HTML5.

3.2.5. PhpMyadmin

Para nuestro proyecto el modelo de base de datos que más se adecua es el modelo relacional, ya que nos permitirá acceder a los distintos elementos de la base de datos a través de las relaciones que establezcamos entre las diferentes tablas mediante el uso de claves primarias y foráneas. Entre las distintas propuestas sobre las que he investigado se encuentran:

- MariaDb
- phpMyadmin
- Oracle
- Mysql

Finalmente he escogido phpMyAdmin (Ilustración 17), ya que es una herramienta de software libre, destinada a manejar la administración de MySQL a través de la Web, además de poder crear toda la información relacionada con las diferentes tablas y las distintas relaciones que existen entre ellas dado que ya he hecho uso de esta herramienta a lo largo de la carrera.

Entre sus principales características cabe destacar:

- Las operaciones de uso frecuente (administración de bases de datos, tablas, columnas, relaciones, índices, usuarios, permisos, etc.) se pueden realizar a través de la interfaz de usuario.
- Capacidad de ejecutar directamente cualquier declaración de SQL.
- Es una herramienta de software libre.

Ilustración 17. Logo phpMyAdmin

3.2.6. Xampp

Hare uso del software XAMPP el cual 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. Es necesario para poder trabajar en la página web de forma local y poder ver los cambios que hacemos en la página web (Ilustración 18).

Ilustración 18. Logo Xampp

3.3. Software Complementario

Además del software que he comentado anteriormente he utilizado otros que están más orientados a la realización del diseño del proyecto o a la ayuda de la realización de la memoria del proyecto.

3.3.1. Visual Paradigm

Una de las herramientas que me ha ayudado a crear todo lo relacionado con la creación de la interfaz de usuario, creación de los casos de uso y los diagramas de secuencia es el software Visual Paradigm (Ilustración 19), que a pesar de ser requerir una licencia se puede obtener a través de los acuerdos software que tiene la Universidad de Jaén para los estudiantes.

Ilustración 19. Logo Visual Paradigm.

3.3.2. GanttProject

Es un programa de código abierto con licencia GPL escrito en Java, cuyo objetivo es la administración de proyectos usando el diagrama de Gantt (Ilustración 20).

Ilustración 20. Logo GanttProject.

4. Capitulo 4. Desarrollo

A partir de este capítulo, comienzo con el desarrollo del proyecto siguiendo el modelo incremental, según el cual, para cada incremento es necesario construir cuatro pasos, un primer paso es el de análisis, en el cual se expondrán los requisitos que debe cumplir ese incremento. El segundo paso es el de diseño, donde se hace un estudio de las soluciones necesarias para afrontar los requisitos dispuestos en la fase anterior de análisis. El tercer paso es el de la implementación del incremento, en esta fase se procederá a crear el código necesario para llevar a cabo el incremento siguiendo los directrices tomadas en la fase de diseño. Y por último la fase de pruebas, donde se realizan una serie de pruebas del resultado obtenido y se comprueba su validez.

Al inicio de cada incremento mostraré una tabla indicando las partes que se llevaran a cabo en dicho incremento, así como pruebas que se realizaran, con el objetivo de hacer más fácil la lectura del documento.

4.1. Primer Incremento

A menudo el primer incremento es el más importante, ya que contiene la parte esencial del proyecto, por lo que en este apartado se expondrían los requisitos básicos que deben cumplir nuestro proyecto.

Como no existe un cliente real, yo mismo hare las veces de cliente y expondré los diferentes objetivos que se debe realizar en cada incremento.

Objetivo hardware: Quiero tener un sistema hardware con el que fichen los trabajadores a través de NFC tanto con tarjetas como llaveros, el sistema debe ser capaz emitir una respuesta visual a la hora de realizar el fichaje y que la comunicación con el servidor se realice tanto por WIFI como por cable ethernet.

Objetivos software: Quiero tener una página web donde cada usuario pueda iniciar sesión con su correo (único) y contraseña y en el que se puedan visualizar las horas de trabajo almacenadas por el módulo de identificación de cada usuario. Cada usuario debe estar asociado a un departamento, que tendrá asociado a su vez un único jefe de Departamento. A su vez deben existir tres tipos de roles, administrador,

jefe de grupo y usuario. Cada uno con diferentes privilegios sobre el sistema, a continuación, se expondrá sus capacidades:

- Administrador: Es el usuario con mayor accesibilidad en el sistema, tiene las opciones de crear y dar de baja a usuarios de cualquier rol. También puede crear grupos de trabajo y asignarlos a usuarios con rol de jefe de grupo. Puede visualizar todos los perfiles de los usuarios que están dados de alta en el sistema y acceder a información relacionada con sus horas de trabajo, nombre, correo y grupos de trabajo al que pertenece.
- Jefe de Departamento: Es el segundo usuario con más accesibilidad en el sistema, sus capacidades son similares a las del administrador, con la diferencia de que a la hora de crear y visualizar usuarios solo podrá hacerlo de aquellos que tengan rol de usuario y que además estén asociados a su grupo de trabajo, sin embargo, no tendrá la opción de crear nuevos grupos ya que esa competencia es solo del administrador del sistema.
- Usuario: Solo podrá acceder a información relacionada con él. Tendrá la opción de modificar su correo electrónico, contraseña y seleccionar una foto de perfil. Además, tendrá la opción de consultar sus horas de trabajo filtrado por días, mes y año.

4.1.1. Análisis

Al ser la primera fase de análisis, veo la necesidad de definir los requisitos tanto funcionales como no funcionales del sistema, que se encargaran de almacenar básicamente lo que el sistema va a hacer. Estos requisitos se han creado a partir de una previa reunión con el cliente, de esta manera se puede obtener de primera mano las necesidades que quiere que su sistema cubra.

4.1.1.1. Requisitos Funcionales

Son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.

En nuestro caso se ha creado una lista con los requisitos que el sistema debe cumplir, centrándonos solo en lo que a la página web respecta y según el tipo de usuario que ha iniciado sesión, ya que todo el análisis y diseño del módulo se realizara en incrementos posteriores.

Lista de requisitos funcionales:

1. El administrador podrá crear nuevos usuarios con cualquier rol y asociarlos a cualquier departamento.
2. Solo el administrador podrá crear nuevos departamentos y asociarlos a cualquier jefe de departamento.
3. Los usuarios con rol jefe de departamento solo podrán crear usuarios con rol usuario y asociarlos a su propio departamento.
4. Solo el administrador del sistema tendrá acceso a toda la información relacionada con los usuarios.
5. Los usuarios con rol jefe de departamento solo podrán ver la información relacionada con los usuarios asociados al suyo.
6. Solo los usuarios dados de alta por el administrador o un jefe de departamento podrán iniciar sesión.
7. Todos los usuarios con rol usuario podrán acceder a información de su perfil, modificar su correo y contraseña, seleccionar una foto de perfil y consultar sus horas de trabajo.
8. El sistema debe ser accesible desde cualquier tipo de dispositivo, por lo que debe tener un diseño responsive.

4.1.1.2. Requisitos No Funcionales

Se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento, es decir, son las restricciones o condiciones que impone el cliente al programa que necesita, por ejemplo, el tiempo de entrega del programa, rendimiento, disponibilidad, el lenguaje o la cantidad de usuarios.

Al igual que con los requisitos funcionales, he elaborado una lista enumerando los requisitos no funcionales aplicables a nuestra página web, como se muestra a continuación:

1. La página web debe estar disponible para su uso las 24 horas del día.
2. El tiempo de respuesta de la información que se desea visualizar en la página web debe ser razonable.
3. Debe asegurar la privacidad del usuario, sin mostrar información que sea ajena al mismo.
4. Debe proporcionar seguridad mediante cifrado de contraseñas y evitar el acceso a información asociada a otros usuarios.

4.1.1.3. Diagrama de Caso de Uso

También hare uso de diagramas de caso de uso, que sirven para crear una descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso. Se utilizan para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. Al ser la primera iteración, creare un caso de uso que servirá de base y que englobara a todas las actividades que se podrán realizar en el sistema, teniendo en cuenta los requerimientos anteriormente descritos por el usuario. De forma que en cada incremento se realizara el análisis, diseño e implementación de cada una de las diferentes actividades.

El acceso a las diferentes opciones dependerá de los privilegios que tenga cada tipo de usuario, como se muestra en la Ilustración 21:

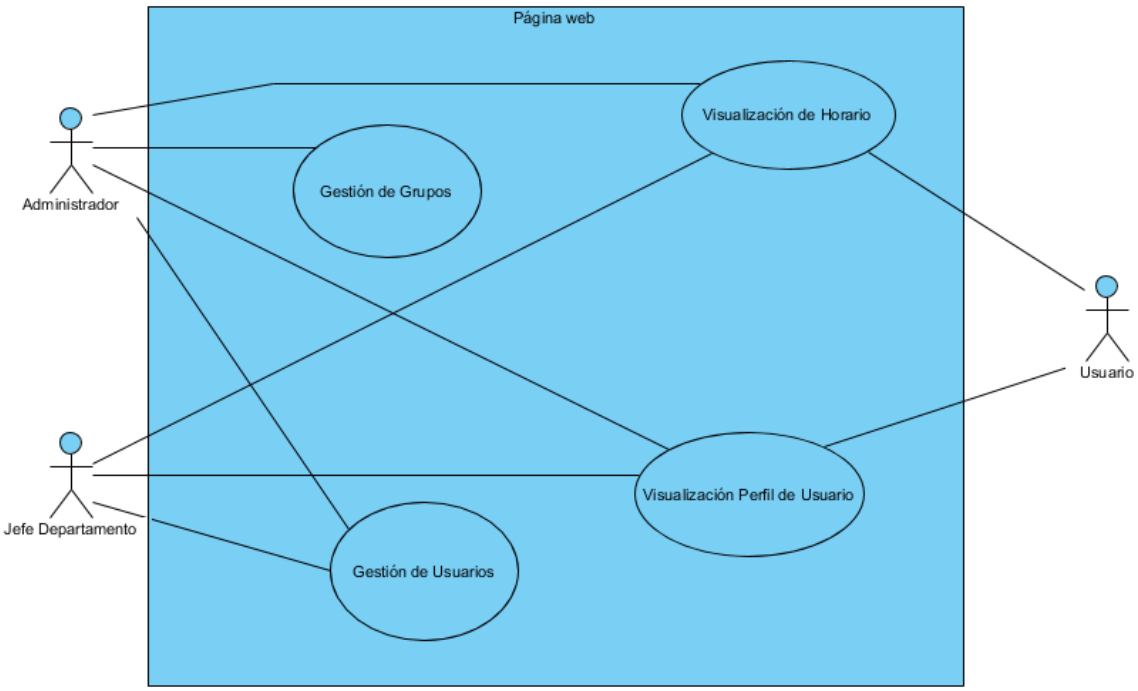


Ilustración 21. Diagrama de Casos de Uso

4.1.2. Diseño

En este primer incremento como he comentado antes, sirve de base del proyecto en general por lo que en esta primera fase de diseño se procederá a la creación de una base de datos que sea capaz tanto de almacenar como de relacionar la información sobre los usuarios, los departamentos a los que pertenecen, los responsables de estos departamentos y las horas de trabajo. También será necesario crear un diagrama de secuencia de la funcionalidad que se desea implementar para este incremento, así como de la selección de una interfaz de usuario apropiada para el proyecto.

4.1.2.1. Base de Datos

A continuación, se expondrá las especificaciones de cada una de las tablas creadas para llevar a cabo el proyecto.

He creado tres tablas:

- Tabla usuario: para almacenar la información de los usuarios.
- Tabla grupo: para almacenar la información sobre los diferentes departamentos existentes en la empresa.
- Tabla horas: para almacenar la información sobre las fechas a la que los usuarios entran y salen de trabajar.

Tabla usuario

Dispone de 13 filas cada una con sus diferentes especificaciones:

- id: identificador único de usuario y clave primaria.
- nombre: nombre del usuario
- primer_apellido: primer apellido del usuario
- segundo_apellido: segundo apellido del usuario.
- DNI: DNI del usuario sin letra.
- grupo: grupo al que pertenece el usuario.
- email: email con el que se registrara el usuario.
- contraseña: contraseña con la que se identificara el usuario para entrar en la web.
- rol: Papel que desempeñará el usuario en la aplicación por defecto el campo rol toma el valor 2. Los roles pueden ser:
    - 0: Para el administrador, el cual podrá acceder a todas las opciones disponibles de la página web (crear usuarios, acceder a perfiles, crear grupos).
    - 1: Para los responsables de grupos (Profesores), podrán disponer de varias opciones, así como la de crear nuevos grupos, crear usuarios y observar los perfiles del usuario asignados a su grupo.
    - 2: Para los usuarios estándar, solo tienen las opciones de consultar su propio horario y editar su propio perfil.
- estado: muestra el estado en el que se encuentra el usuario, es decir, si se encuentra actualmente trabajando en un grupo.
- imagen: imagen asociada a cada perfil de usuario, por defecto está asignada una foto estándar.

Tabla grupo

La siguiente tabla es la tabla grupos, la cual necesitaremos para saber que grupos existen en la empresa y que usuarios están asociados a cada grupo, para ello utilizaremos:

- id_grupo: identificador único de grupo que será a su vez la clave primaria.
- nombre_grupo: nombre del grupo es único de manera que no puedan existir dos grupos con el mismo nombre.
- id_usuario: identificador del usuario responsable del grupo, solo pueden hacerse cargo usuario con un rol distinto a 2, también es la clave foránea de la tabla.

Tabla Horas

En esta tabla se debe almacenar la información sobre las horas de entrada/salida de los trabajadores de la empresa, por lo que necesitaremos los siguientes atributos en la tabla:

- id_hora: identificador único de la tabla hora, a su vez clave primaria de la tabla.
- id_usuario: identificador de usuario asociado a esa hora, es la clave foránea de la tabla.
- hora: hora en formato timestamp en la que se produce el fichaje.
- estado: Nos indica si se trata de una entrada o de una salida.

Tras tener diseñados los atributos de cada tabla, es necesario tener un modelo de las relaciones que existen entre ellas, de manera que a través de las diferentes conexiones seamos capaces de acceder a la información que necesitamos, como se muestra en la siguiente imagen.

4.1.2.2. Diagrama de Secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. A menudo es

útil para complementar a un diagrama de clases. En este incremento se desarrollará la funcionalidad básica de acceder a la página web con nuestro email y contraseña como se representa en la Ilustración 22.

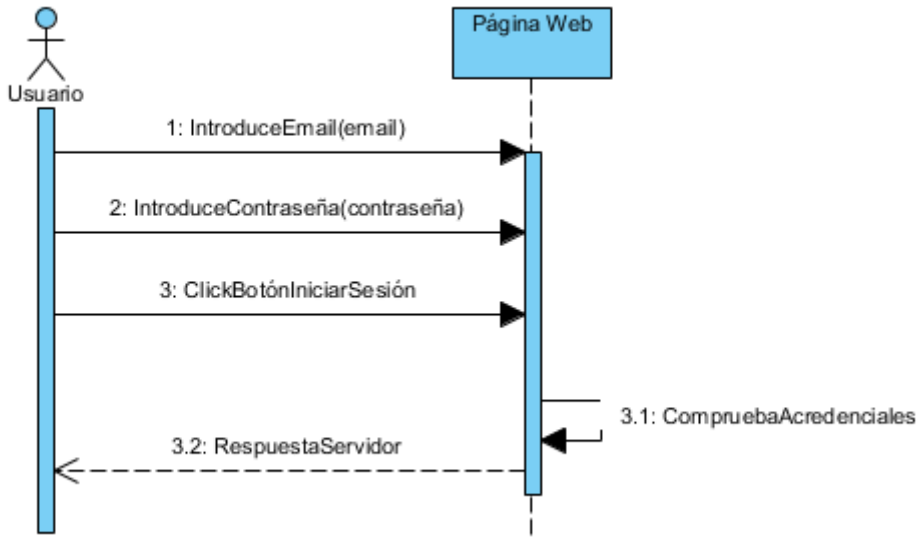


Ilustración 22. Diagrama de Secuencia de inicio de sesión.

4.1.2.3. Interfaz de Usuario

La interfaz de usuario es la que permite al usuario comunicarse con el dispositivo, esta determina en gran medida la percepción e impresión que el usuario posee de una aplicación. El usuario no está interesado en la estructura interna de una aplicación, sino en cómo usarla, es por ello que se debe procurar que las interfaces de usuario sean usables e intuitivas. Debido a estas razones, veo la necesidad de estudiar de antemano los principios básicos para garantizar que un sistema sea usable.

La usabilidad es la medida en la que un producto puede ser usado por determinados usuarios para conseguir unos objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso dado.

Un software es usable si es:

- Fácil de aprender: Permite realizar las tareas rápidamente y sin errores.

- Fácil de utilizar: el usuario usa la herramienta de una manera más natural.

Una vez entendido como se debe crear una interfaz, me basare en las técnicas de prototipado para la creación de la misma. Los prototipos son documentos o diseños que simulan o tienen implementadas partes del sistema final. Son herramientas útiles para hacer participar al usuario en el diseño y poder evaluarlo en fases tempranas. Existen diferentes tipos de prototipos entre los que destacan:

- Flowchart: Representación gráfica de las acciones y decisiones extraídas de la narrativa.
- Texto procedural: Descripción paso a paso de las acciones del usuario y las respuestas del sistema
- Storyboard: Es una narración gráfica de una historia en cuadros consecutivos. El storyboard nos permite indicar los enlaces a diferentes páginas.
- Prototipo de papel: Este tipo de prototipo se basa en la utilización de papel, tijeras, lápiz o instrumentos que se puedan utilizar para describir un diseño en un papel. Este sistema nos da una gran velocidad y flexibilidad.

En mi caso voy a hacer uso de la creación de escenarios mediante el uso de Storyboard, ya que es una herramienta fácil y rápida de utilizar y en la que el usuario se sentirá cómodo. Creare todas las vistas de nuestro sistema, aunque la mayoría de ellas se implementaran en los sucesivos incrementos y pasare a una breve explicación de las acciones disponibles en cada una de ellas.

La Ilustración 23 es la vista de inicio de sesión y a partir de la cual se podrá acceder a nuestro sistema.

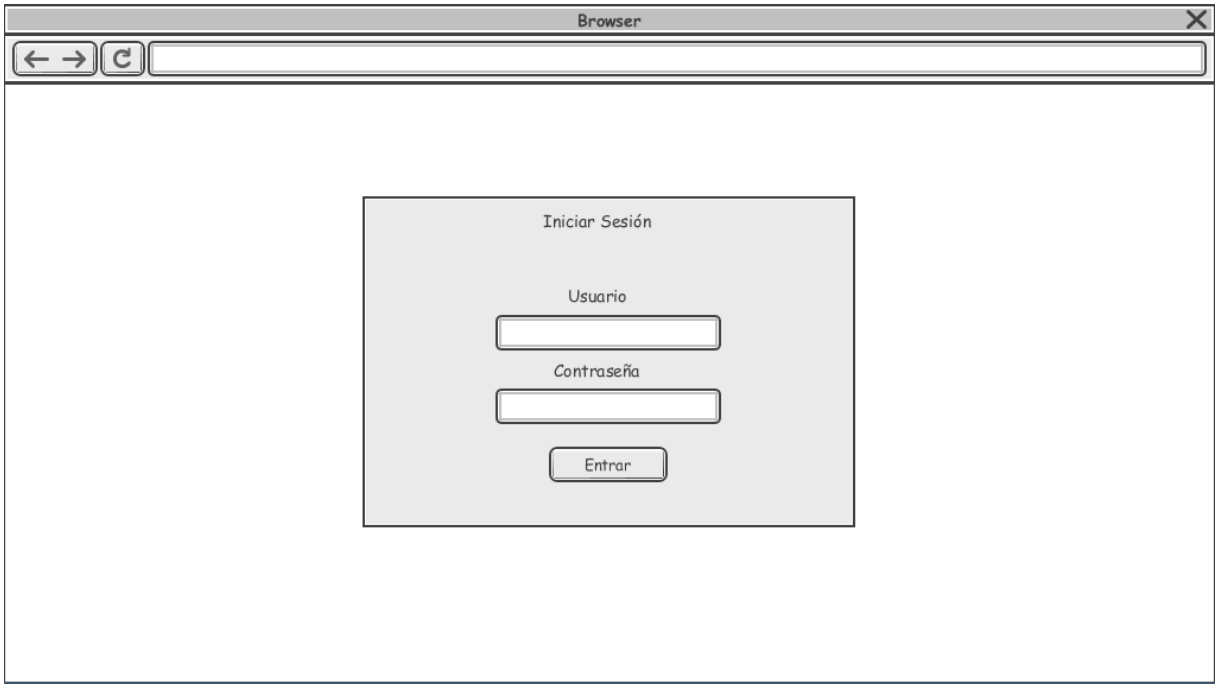


Ilustración 23. Vista inicio sesión.

La Ilustración 24 será la vista principal a la que accederán los usuarios con mayor accesibilidad al sistema, donde aparecerá una tabla con los usuarios a los que puede acceder el usuario registrado.

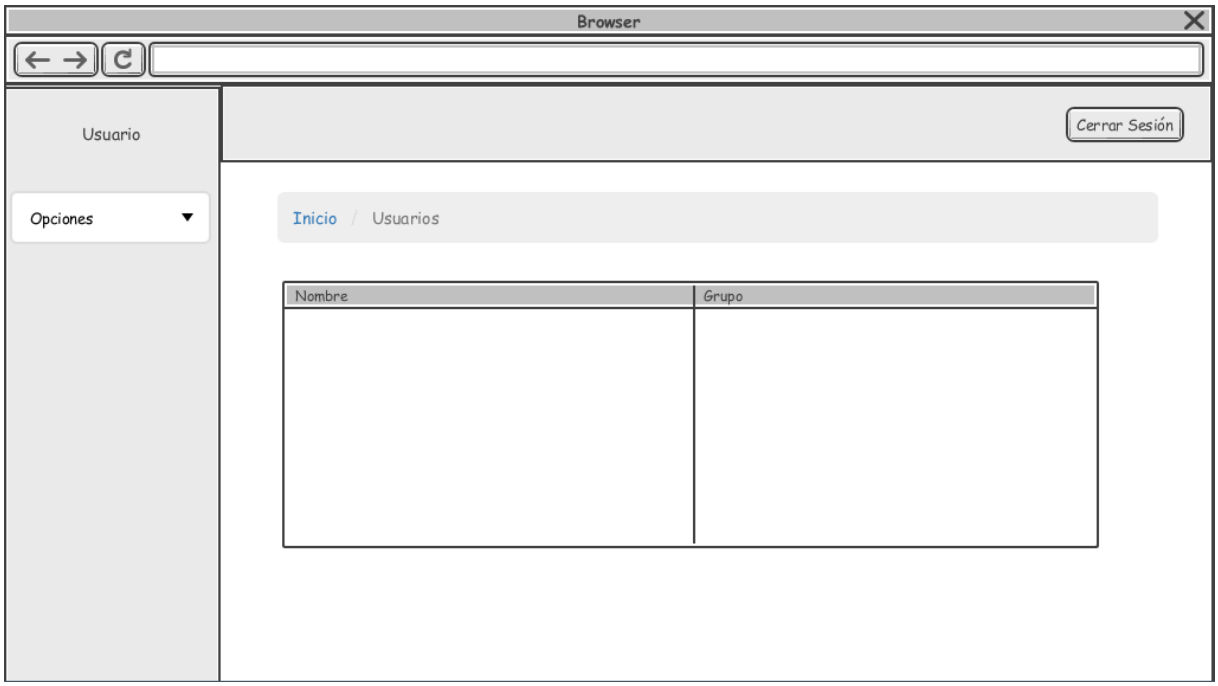


Ilustración 24. Vista principal de administrador y jefe departamento.

La Ilustración 25 será la principal para los usuarios con menor accesibilidad y en ella aparecerá la información sobre las horas fichadas por dicho usuario y contará con un buscador para poder buscar entre un intervalo de fechas concreto. Además, esta vista estará disponible para el administrador y para los jefes de departamento para consultar las horas de los usuarios a los que tienen accesibilidad.

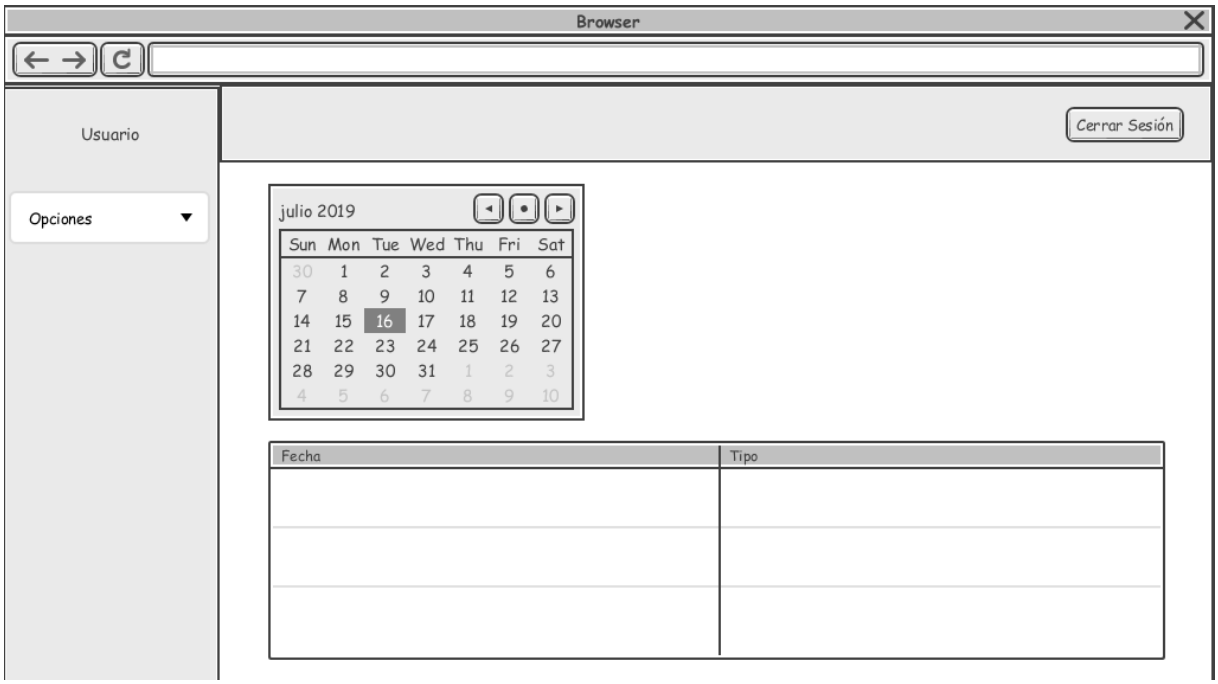


Ilustración 25. Vista principal usuario / consulta horaria.

En la Ilustración 26 se muestra la vista que solo es accesible para el administrador y podrá ver todos los departamentos creados y sus responsables.

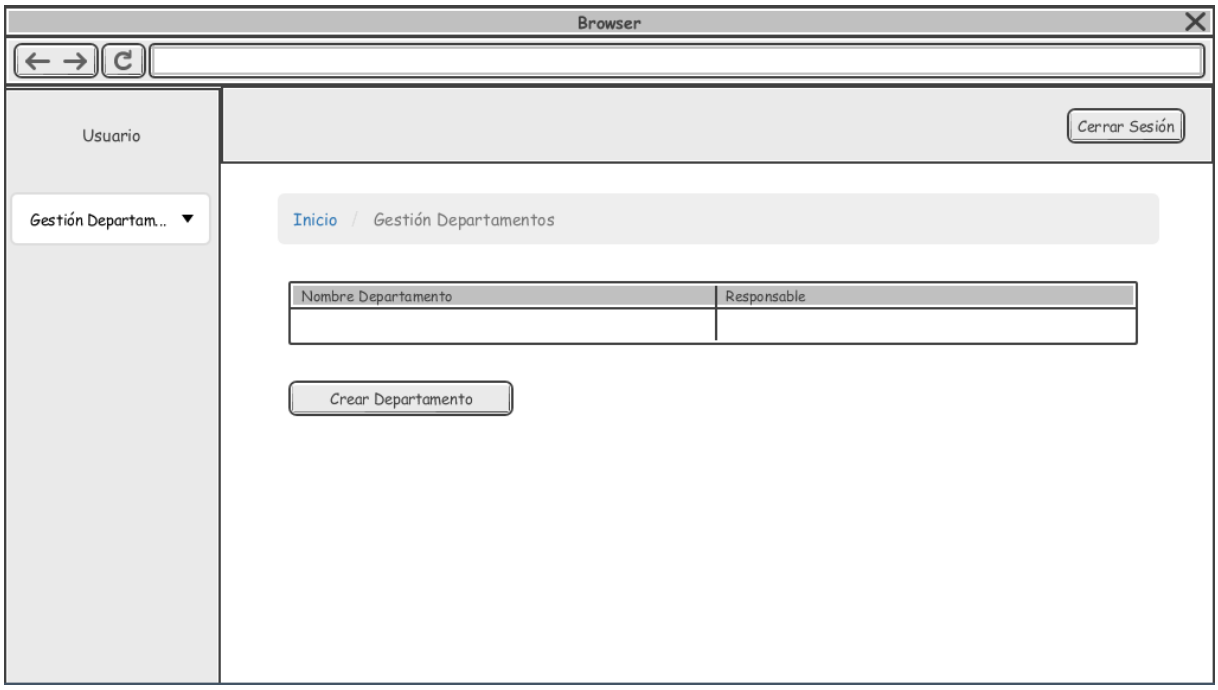


Ilustración 26. Vista Gestión Departamentos.

En la Ilustración 27 aparece la vista que solo es accesible por el administrador y en ella podrá dar de alta nuevos grupos y asociarlos a los jefes de departamento.

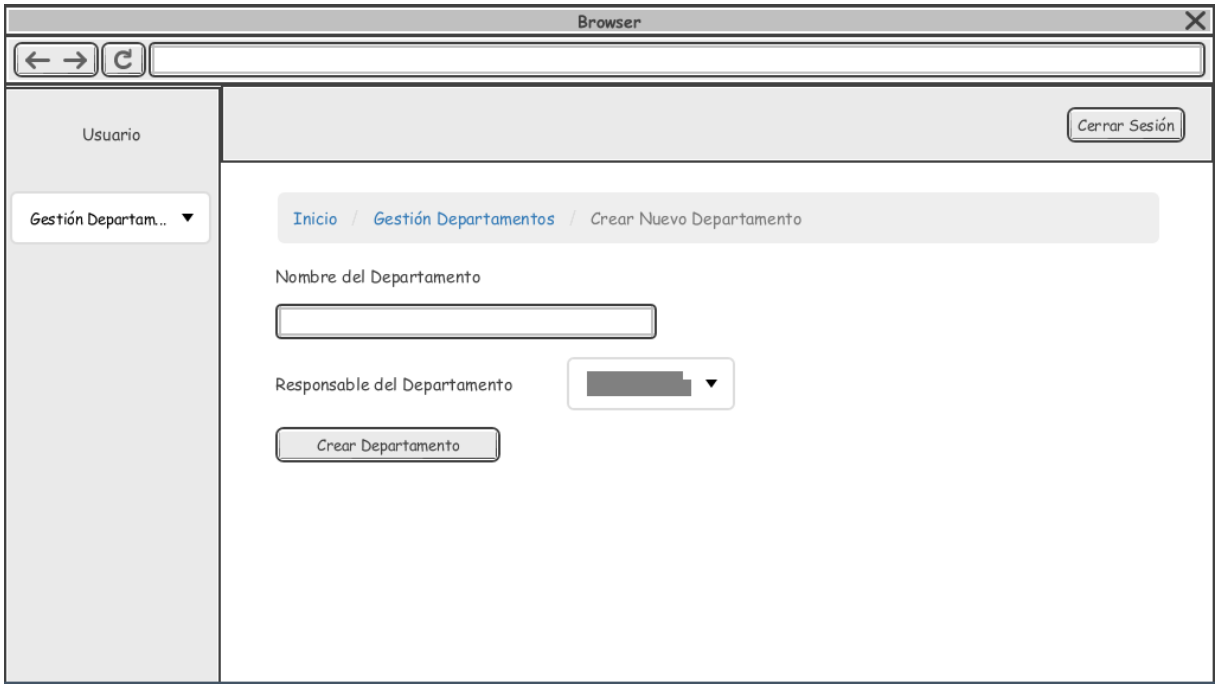


Ilustración 27. Vista crear departamento.

La vista para crear usuario (Ilustración 28) será accesible para los usuarios con mayor accesibilidad y en ella crearan nuevos usuarios con una accesibilidad menor a la de ellos mismos.

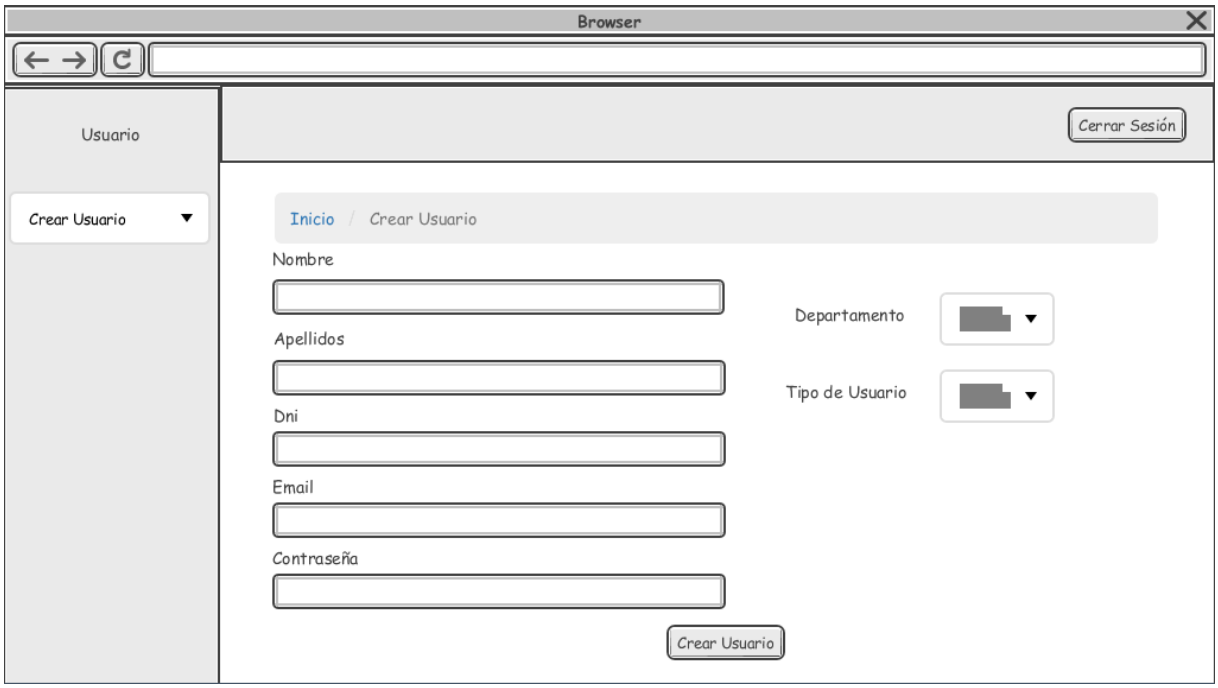


Ilustración 28. Vista crear usuario.

Tanto las vistas de perfil de usuario (Ilustración 29), como las de modificar el perfil (Ilustración 30) serán accesibles por todos los usuarios que puedan acceder al sistema.

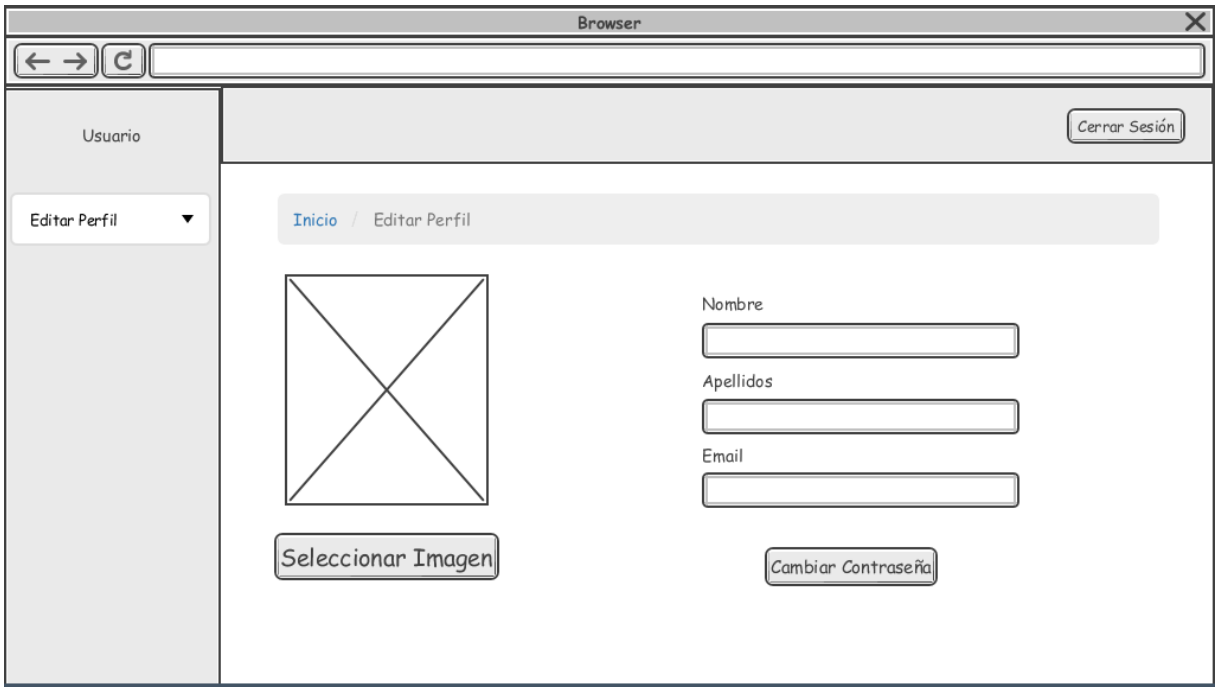


Ilustración 29. Vista perfil de usuario.

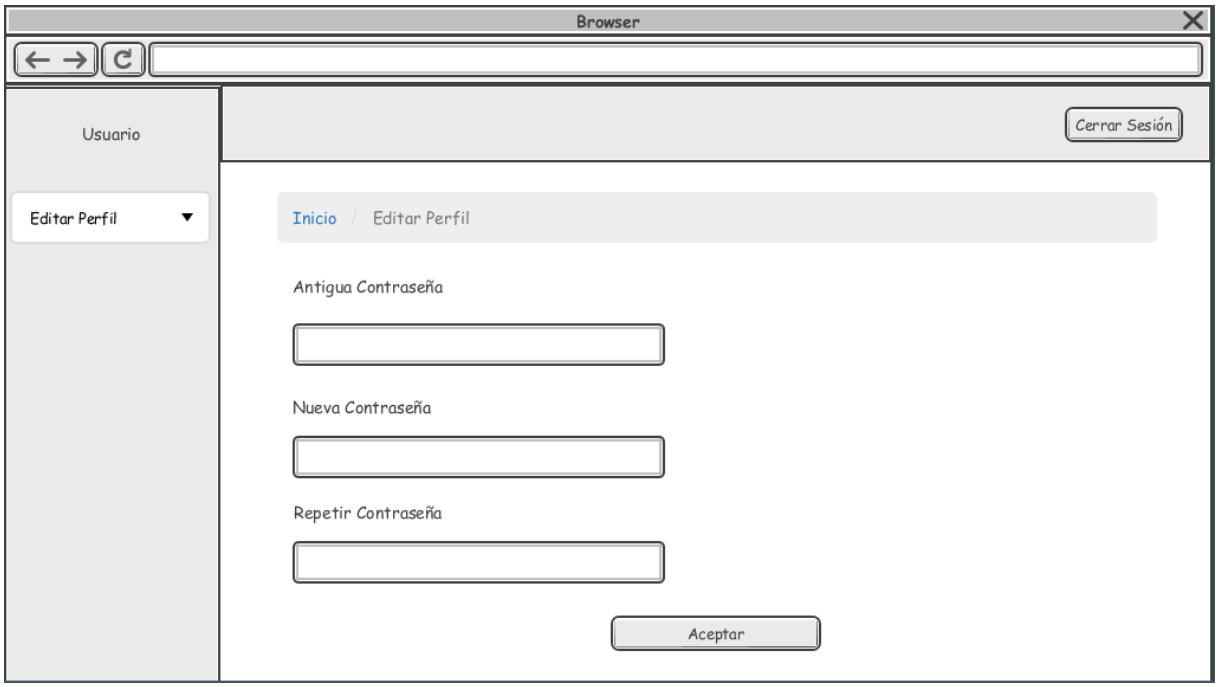


Ilustración 30. Vista cambiar contraseña.

4.1.3. Implementación

Al igual que en el diseño de este incremento, dividiré la implementación en dos partes, una para la base de datos y otra para la implementación de la página web.

4.1.3.1. Base de Datos

Tras la creación de las diferentes tablas y especificación de los atributos he obtenido como resultado el mostrado en la Ilustración 31.

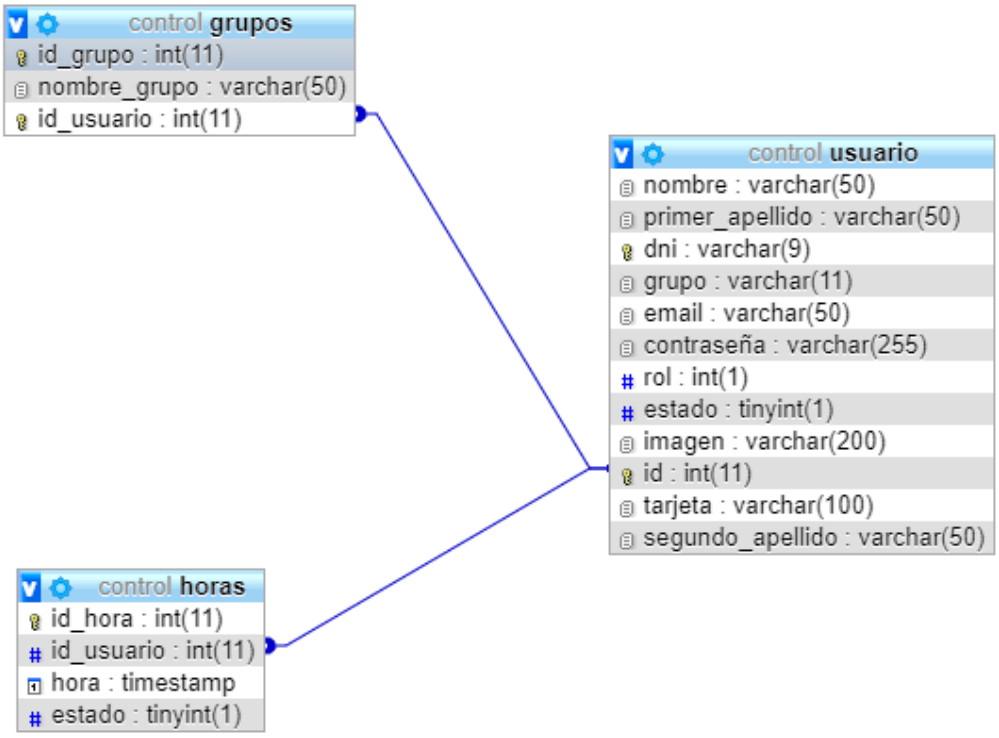


Ilustración 31. Resultado BBDD.

4.1.3.2. Página web

Utilizare el patrón de diseño Modelo-Vista-Controlador (Ilustración 32), que es un patrón de arquitectura de software, en este modelo se separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

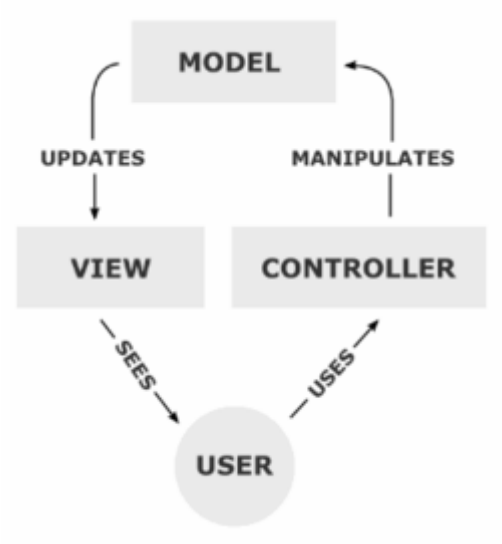


Ilustración 32. Patrón Modelo-Vista-Controlador

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Llevar un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador es responsable de:

- Recibir los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las

vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()".

Las vistas son responsables de:

- Recibir datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Arquitectura:

El sistema de identificación hace uso de la arquitectura cliente-servidor (Ilustración 33) que es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

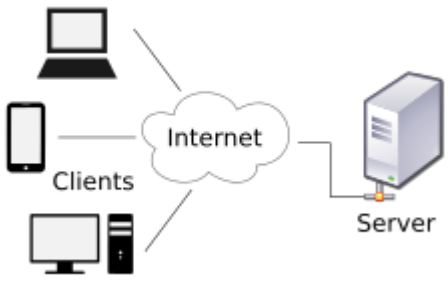


Ilustración 33. Arquitectura Cliente-Servidor

En este sistema los clientes a la hora de fichar mandan un mensaje a nuestro servidor con la fecha y hora correspondientes a la del fichaje.

Lo primero que haré antes de comenzar la implementación es instalar el software Xampp para poder trabajar como he explicado en el apartado de tecnologías de forma local. Su instalación es muy sencilla solo hay que escoger la ruta donde se desea instalar. Tras la instalación de este software es hora de instalar el framework code igniter que será nuestro plano para moldear el modelo de página web descargado. Su instalación debe realizarse en la carpeta htdocs (imagen), donde colocaremos una carpeta ci en la cual se encuentra toda la información de codeIgniter como se ve en la Ilustración 34 y la Ilustración 35.

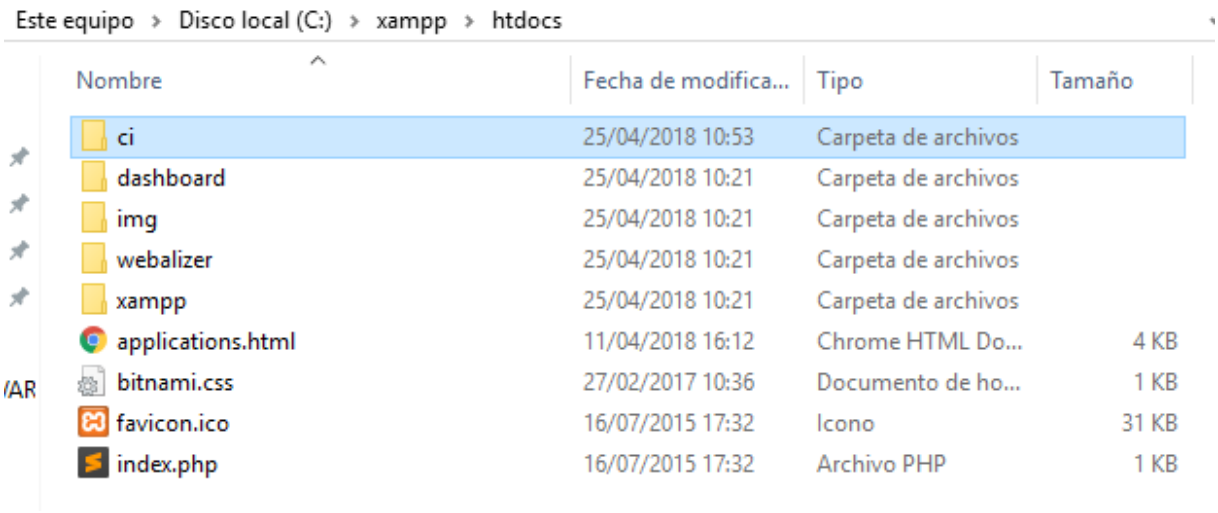


Ilustración 34. Carpetas htdocs.

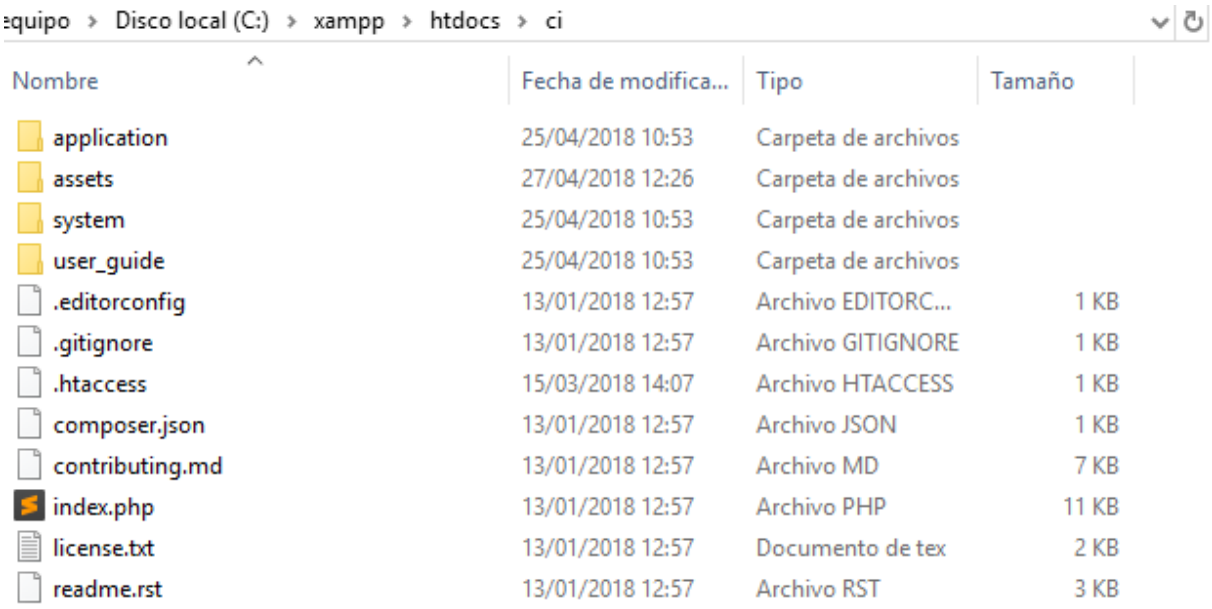


Ilustración 35. Carpetas del proyecto.

Tras la debida instalación de los diferentes softwares, es necesario crear los ficheros donde se almacenarán los modelos para poder obtener la información que se almacena en la base de datos a través de consultas jquery. Hay que crear un modelo por tabla existente en la base de datos, por ello crearemos:

- Usuario_model: Para las consultas sobre información de usuarios.
- Grupo_model: Para las consultas sobre información de grupos.
- Horas_model: Para las consultas sobre información de horas.

Ahora es el turno de identificar que partes del estilo escogido formarán parte de la cabecera o header, y que parte lo hará del pie de página o footer. Estas vistas que siempre estarán presentes durante la navegación de la página web las guardare en una carpeta llamada layout dentro de la carpeta views.

El Header estará formado por la barra de navegación lateral y superior. En la barra superior aparecerá el nombre del usuario logueado y la opción de cerrar sesión, mientras que en el lateral aparecerán las distintas opciones que tienen los usuarios. El resultado es el mostrado en la Ilustración 36.

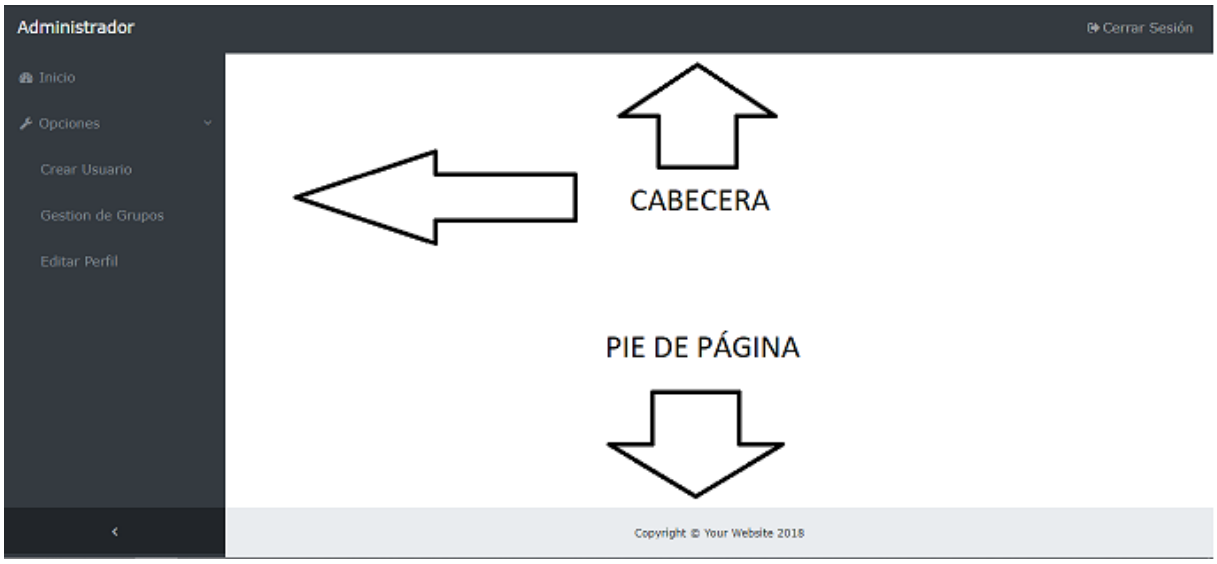


Ilustración 36. Header y Footer.

El siguiente paso es crear la vista login en la carpeta views de codeIgniter con los elementos necesarios para poder realizar un registro de usuario correcto. Para ello utilizaremos el login del modelo de página web seleccionado, con la excepción de que en nuestra página web los usuarios no pueden darse de alta por si solos, por lo que la opción de crear usuario desaparecerá del menú, obteniendo como resultado el mostrado en la Ilustración 37.

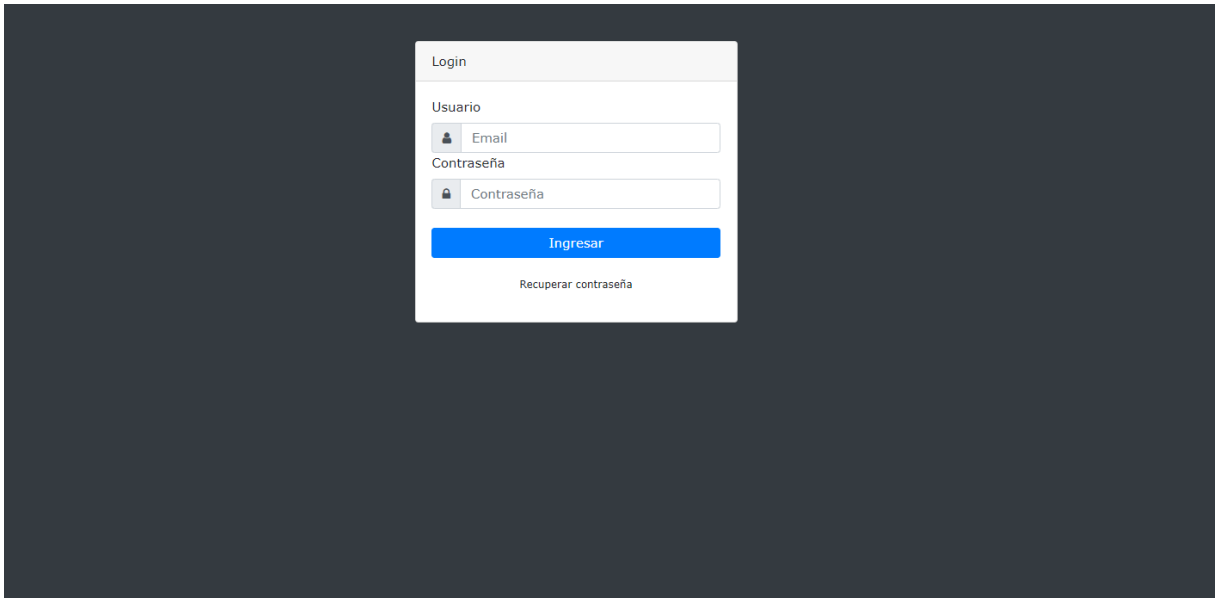


Ilustración 37. Resultado Login.

El último paso y más extenso, es el de crear una validación de los campos correo electrónico y usuario de manera correcta y segura. Por ello codeIgniter dispone de una librería llamada Bcrypt que hace imposible que alguien pueda acceder a la versión en texto de la contraseña si nuestra base de datos se ve comprometida. La librería dispone de dos métodos que serán los necesarios para realizar operaciones con las citadas contraseñas:

- Hash_password(passaword): Este método devuelve la encriptación de una contraseña pasada como parámetro.
- Check_password(contraseña, password): Este método devuelve true si la contraseña pasada por parámetro se corresponde a la password obtenida de la base de datos.

También hare uso de la librería session, cuyo funcionamiento es semejante al que usan las cookies de una página web, almacenando información sobre el usuario hasta que este cierra sesión en la página. Esta librería dispone de la función user_data(usuario) donde usuario contiene la información básica del usuario loguedao, en nuestro caso:

- id
- nombre
- apellidos
- rol
- email
- grupo
- DNI
- estado
- imagen

Gracias a esto, podemos consultar la información del usuario en cualquier momento sin tener que hacer una consulta a la base de datos, haciendo así la navegación entre páginas más eficiente, simplemente hay que llamar al método $this->session->userdata(información_a_consultar).

Continuando con la validación, es necesario crear un controlador que llamaremos inicio que se encargue de validar los campos del formulario login, como hemos explicado en la primera interacción el controlador se comunica tanto con la vista como con los modelos para obtener y mandar información. Por ello la función índex de este fichero recibe los campos usuario y contraseña, y a partir de aquí realiza diferentes comprobaciones, una detrás de otra para mostrar determinados errores por pantalla:

1. La primera comprobación es saber si ya hay una sesión iniciada en la página web, de forma que si no se ha cerrado la sesión aparezca la información de ultimo usuario registrado, si se ha cerrado sesión correctamente se pasa a la siguiente comprobación.
2. Comprueba si existe el correo electrónico en la base de datos, de no ser asi muestra por pantalla que el usuario no existe.
3. Comprueba si la contraseña introducida corresponde con la contraseña obtenida de la base de datos con la función check_password, de no ser asi muestra por pantalla que la contraseña es errónea.

Cuando la validación tiene éxito, se crea la variable user_data que nos facilita la librería session como se ha comentado anteriormente y se rellena con la información del usuario y a su vez se llama a la página principal de la página web, que mostraría información básica del usuario. En la Ilustración 38 y la Ilustración 39 se puede ver el código asociado.

```php
/**
 * Comprueba el usuario y la contraseña y muestra la vista principal de la web
 * si los credenciales son correctos o redirecciona al login
 *
 * @access public
 * @param
 * @return
 */
public function index()
{
  if(!$this->session->userdata('nombre')){
   if(isset($_POST['email']) && isset($_POST['password'])){
            //$hash=$this->bcrypt->hash_password($_POST['password']);
    if($this->usuario_model->buscarEmail($_POST['email']) > '0'){
      $hash=$this->usuario_model->getPasswordEmail($_POST['email']);
           if($this->bcrypt->check_password($_POST['password'], $hash)){   // La variable $ExisteUsuarioyPassoword recibe valor TRUE si el
           usuario existe y FALSE en caso que no. Este valor lo determina el modelo.
              //$name=$this->usuarios_model->obtenerUsuario($_POST['email']);
             $result=$this->usuario_model->getUsuarioEmail($_POST['email']);
             $user = array(
               'id' => $result->id,
               'nombre'=> $result->nombre,
               'primer_apellido' => $result->primer_apellido,
               'segundo_apellido' => $result->segundo_apellido,
               'rol' => $result->rol,
               'email' => $result->email,
               'grupo' => $result->grupo,
               'dni' => $result->dni,
               'estado' => $result ->estado,
               'imagen' => $result->imagen
             );

             $this->session->set_userdata($user);

             if($result->rol != 2){
                 $data['usuario']=$this->obtenerTabla();//$this->usuario_model->getUsuarios($_SESSION['grupo']);
                 $this->load->view('layout/header');
                 $this->load->view('tablaUsuarios',$data);
                 $this->load->view('layout/footer');

               }else{
                $this->load->view('layout/header');
                $this->horario($this->session->userdata('id'));
                $this->load->view('layout/footer');
              }
```

Ilustración 38. Lógica inicio sesión 1

```php
           }else{   //Si la contraseña no esta bien introducida
            echo "<script>alert('Contraseña incorrecta');</script>";
              redirect('','refresh');   //   Lo regresamos a la pantalla de login y pasamos como parámetro el mensaje de error a presentar en
              pantalla
            }
       }else{ //Si no logró validar
        echo "<script>alert('El correo no existe');</script>";
        redirect('','refresh');

      }

    }else{
      $this->load->view('login');

    }

  }else{

    if($_SESSION['rol'] != 2){
          $data['usuario']=$this->obtenerTabla();//$this->usuario_model->getUsuarios($_SESSION['grupo']);
          $this->load->view('layout/header');
          $this->load->view('tablaUsuarios',$data);
          $this->load->view('layout/footer');

        }else{
         $this->load->view('layout/header');
         $this->horario($this->session->userdata('id'));
         $this->load->view('layout/footer');
       }

     }

   }
```

Ilustración 39. Lógica inicio sesión 2.

4.1.4. Pruebas

En la fase de pruebas de este incremento el usuario deberá de ser capaz de iniciar sesión utilizando un correo y una contraseña que previamente han sido introducidos de forma manual en la base de datos, ya que aún no está creada la funcionalidad de crear usuario a través de la página web.

El resultado es el que se puede apreciar en la Ilustración 40 e Ilustración 41.

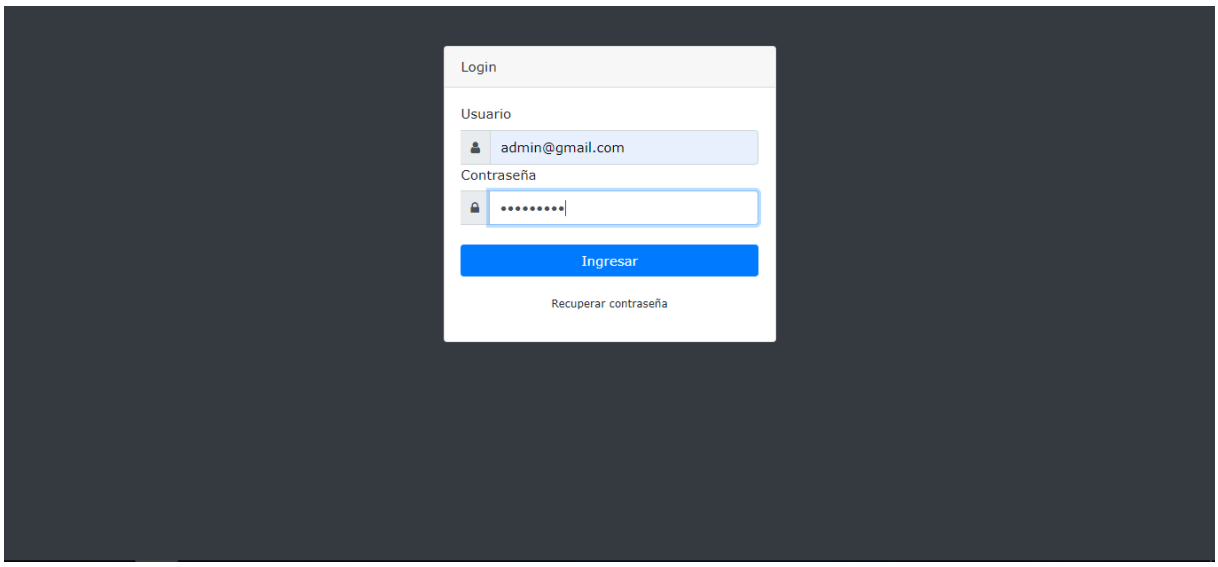


Ilustración 40. Resultado Pruebas 1.

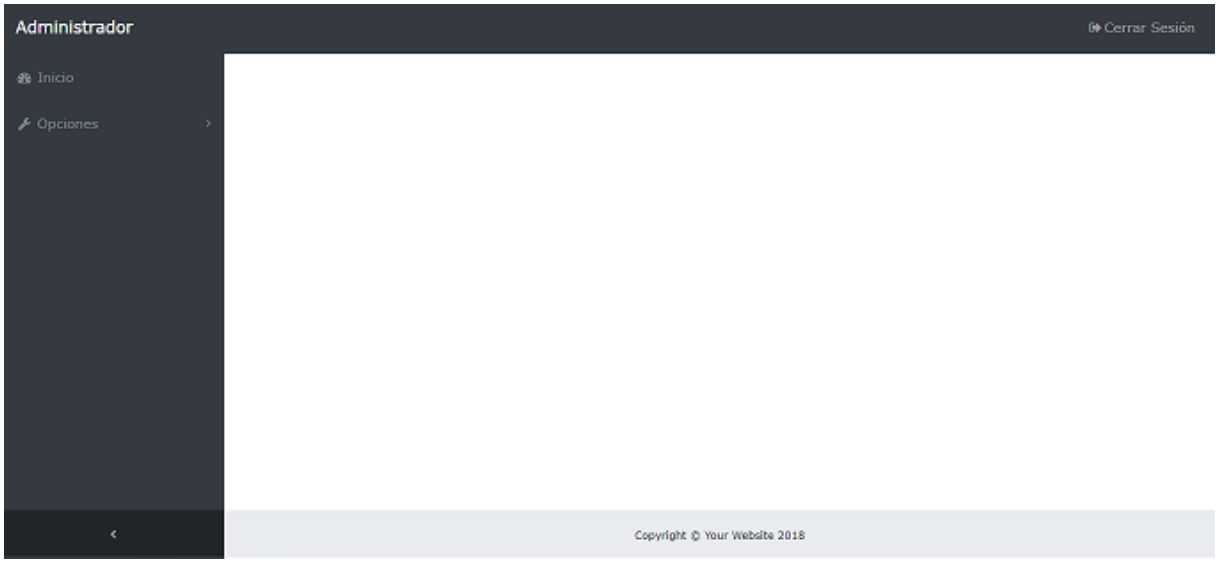


Ilustración 41. Resultado pruebas 2.

4.2. Segundo Incremento

4.2.1. Análisis

En este segundo incremento procederé como se ve en la planificación, a la creación de todas las vistas y acciones relacionadas con la gestión de usuarios.

4.2.1.1. Caso de Uso

En la Ilustración 42, se muestra los casos de uso relacionados con la gestión de usuarios, en la cual se puede apreciar que tanto el administrador como el jefe de departamento comparten la opción tanto de crear nuevos usuarios como de darlos de baja del sistema, la única diferencia es que solo el administrador tiene la opción de asignar el rol al usuario creado, ya que el jefe de departamento solo puede crear usuarios con los privilegios más bajos, es decir, usuarios normales. En cuanto al resto de opciones todos los usuarios podrán iniciar sesión, cerrar sesión y modificar su perfil de usuario.

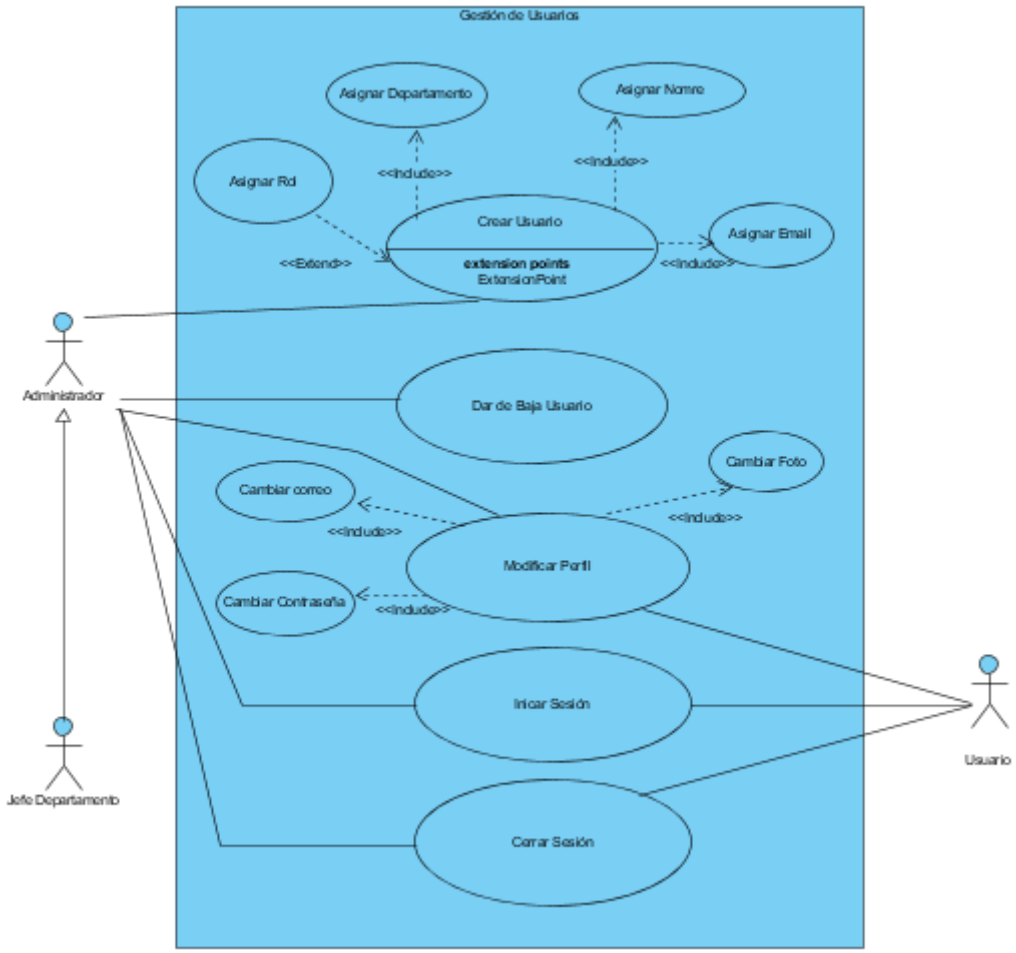


Ilustración 42. Gestión de usuarios

4.2.2. Diseño

4.2.2.1. Diagrama de Secuencia

A continuación, mostrare el diagrama de secuencia de la creación de un usuario, donde se muestra los diferentes pasos que se debe realizar para llevar a cabo la operación.

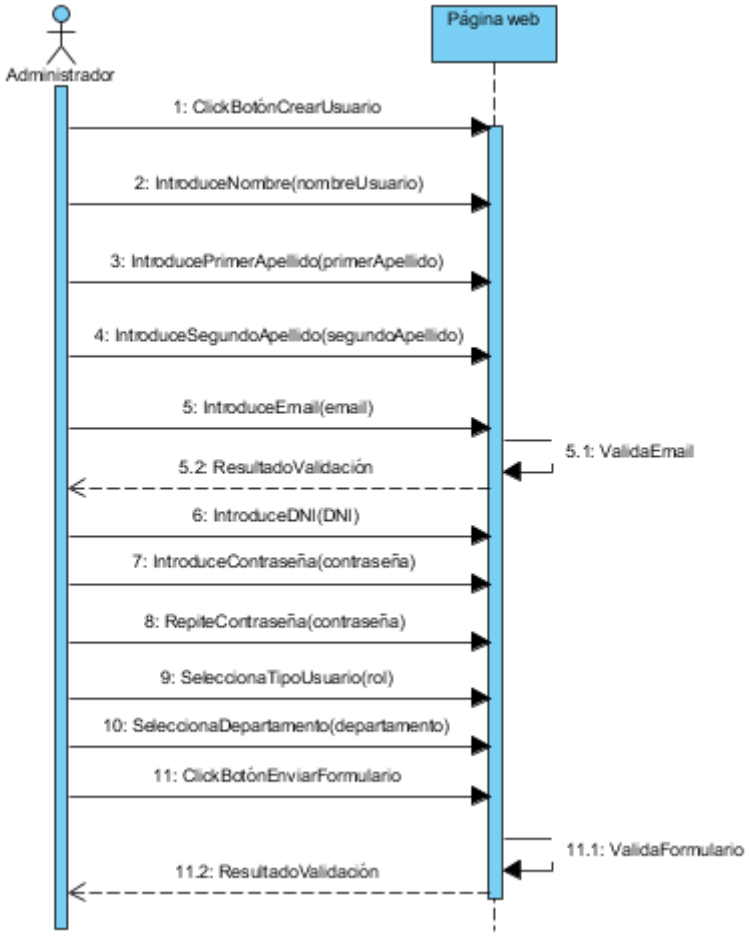


Ilustración 43. Diagrama de secuencia de crear usuario.

El diagrama de secuencia asociado a iniciar sesión ya se ha explicado anteriormente en la Ilustración 22.

4.2.2.2. Interfaz de Usuario

En cuanto a la interfaz de usuario seguiré el creado en el incremento anterior para la vista de perfil y la vista de crear usuario. (Ilustración 28, Ilustración 29)

La funcionalidad de dar de baja a un usuario, no se creará hasta tener creado el módulo de identificación, debido a que está directamente asociada a dicho módulo, aunque visualmente si aparecerá el botón.

4.2.3. Implementación

Empezare con la implementación de la vista de perfil de usuario. El primer paso es crear la carpeta en la que se guardaran las vistas relacionadas con la vista que contiene información sobre el perfil. Para ello en la carpeta ci->application->views creare una carpeta llamada perfil y dentro de ella un archivo php llamado vistaPerfil.php donde se creará el estilo de la vista. El siguiente paso es crear en el modelo usuario los métodos correspondientes para poder realizar una actualización de contraseña y correo electrónico en la base de datos, así como dos métodos para obtener esta información de la base de datos y comunicársela al controlador que la solicite.

Para terminar la organización del usuario se mostrará con la siguiente disposición como puede verse en la Ilustración 44, cada número está asociado a su correspondiente campo en la página web:

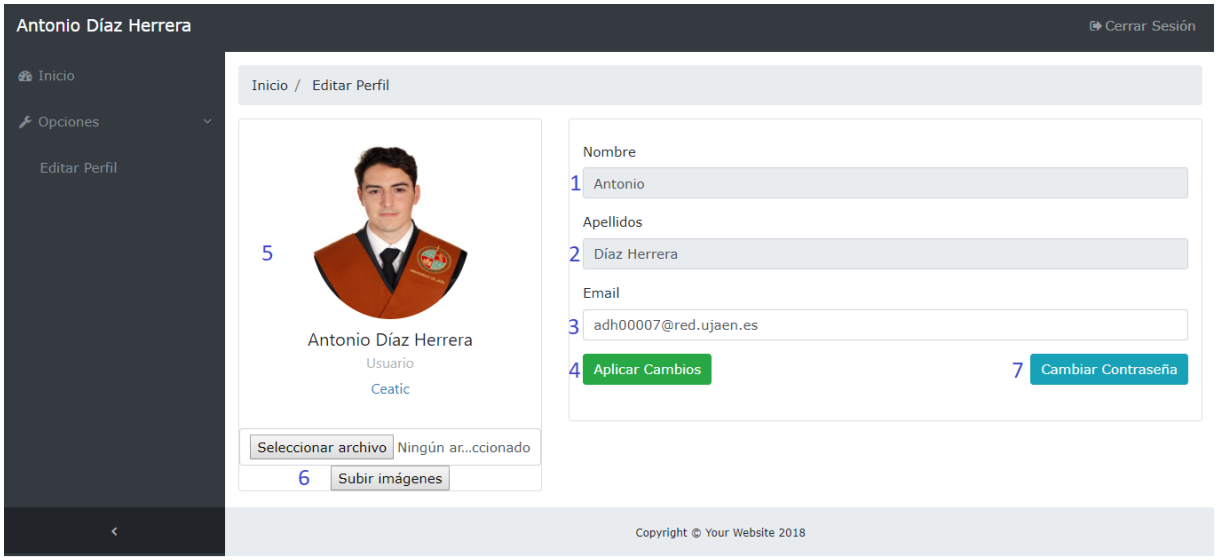


Ilustración 44. Vista Perfil Usuario.

1. Nombre del usuario: No se puede modificar, marcado visualmente.
2. Apellidos del usuario: No se puede modificar, marcado visualmente.
3. Correo del usuario: Puede cambiarse por otro diferente, pero se comprobará que el email introducido no exista ya en la base de datos y que este cumpla los requisitos de ser un email correcto.
4. Aplicar Cambios: Si se modifica el correo electrónico es necesario guardar los cambios pulsando el botón número 4, de forma que

aparecerá un mensaje por pantalla preguntando si deseamos realizar la operación, si aceptamos se le mandará la información al controlador inicio donde he creado un método que comprueba si el texto escrito no supera los 50 caracteres y si el correo ya existe en la base de datos, de ser así nos aparecerá un mensaje por pantalla mostrando un error. Sin embargo, si la operación es correcta realizaremos un cambio en la base de datos del correo y cambiaremos la información guardada en user_data(email) para poder visualizar la nueva información, como resultado aparecerá por pantalla un mensaje de que la operación ha tenido éxito (Ilustración 45).

```php
/**
 * Se encarga de validar si el email introducido es valido y
 * muestra un mensaje confimacion de la operacion
 *
 * @access public
 * @param
 * @return
 */

public function editarEmail(){
  $this->form_validation->set_rules('email','email','max_length[50]|valid_email');


  if ($this->form_validation->run($this)!=FALSE){
   if($this->usuario_model->buscarEmail($_POST['email']) == 0){
    $this->usuario_model->cambiarEmail($_POST['email'],$this->session->userdata('id'));
    $this->session->unset_userdata ( 'email' );
    $this->session->set_userdata ( 'email' ,$_POST['email']);
    $data["error"]="";
    $this->load->view('layout/header');
    echo "<script>alert('Correo cambiado correctamente');</script>";
    redirect('inicio/vistaPerfil','refresh');
    $this->load->view('layout/footer');

  }else{

    $data["error"]="";
    $this->load->view('layout/header');
    echo "<script>alert('Esa direccion de correo ya existe');</script>";
    redirect('inicio/vistaPerfil','refresh');
    $this->load->view('layout/footer');
  }

}else{
  $data["error"]="";
  $this->load->view('layout/header');
  echo "<script>alert('Correo invalido');</script>";
  redirect('inicio/vistaPerfil','refresh');
  $this->load->view('layout/footer');

}


}
```

Ilustración 45. Función editar email.

5. Cambiar Contraseña: Cuando se pulsa el botón aparece un modal (Ilustración 46), con tres campos, el primero para introducir la actual contraseña para asegurarnos que el cambio lo está realizando el propietario de la cuenta, y los dos siguientes para modificar dicha

contraseña. Una vez aceptada la operación se manda esta información al controlador Inicio donde hay un método llamado editarPassword() que recibe esta información y comprueba:

a. Que la contraseña del primer campo sea la misma a la de la base de datos.
b. Que la nueva contraseña este entre 6 y 12 caracteres.
c. Que los campos nueva contraseña y repetir contraseña coincidan.

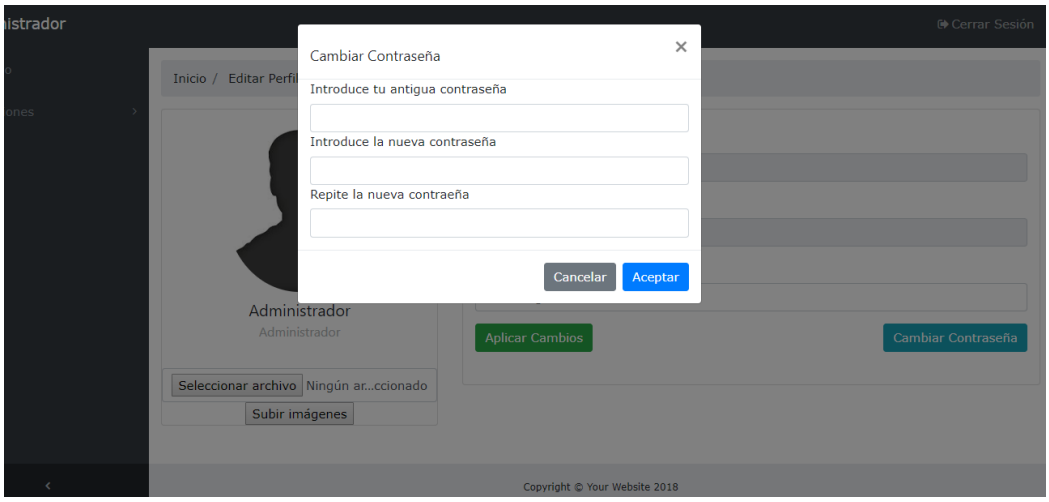


Ilustración 46. Resultado Cambiar Contraseña.

En la Ilustración 47 puede observarse el código asociado a la funcionalidad explicada anteriormente.

```php
/**
 * Valida el formulario para cambiar la contraseña y muestra un mensaje
 * informando de la operacion
 *
 * @access public
 * @param
 * @return
 */

public function editarPassword(){

  $this->form_validation->set_rules('newPassword','newPassword','min_length[6]|max_length[12]');
  $this->form_validation->set_rules('confPassword','confPassword','matches[newPassword]');

  if ($this->form_validation->run($this)!=FALSE){
    if($this->bcrypt->check_password($_POST['password'], $this->usuario_model->getPasswordEmail($this->session->userdata('email')))){
      $hash=$this->bcrypt->hash_password($_POST['newPassword']);
      $this->usuario_model->cambiarContraseña($hash,$this->session->userdata('id'));
      $data['error']="";
      $this->load->view('layout/header');
      echo "<script>alert('Contraseña cambiada correctamente');</script>";
      redirect('inicio/vistaPerfil','refresh');

      $this->load->view('layout/footer');

    }else{

      $data['error']="";
      $this->load->view('layout/header');
      echo "<script>alert('Contraseña actual erronea');</script>";
      redirect('inicio/vistaPerfil','refresh');

      $this->load->view('layout/footer');

    }
  }else{

    $data['error']="";
    $this->load->view('layout/header');
    echo "<script>alert('Las contraseñas no son validas');</script>";
    redirect('inicio/vistaPerfil','refresh');

    $this->load->view('layout/footer');

  }
```

Ilustración 47. Función Editar Contraseña.

6. Imagen del usuario: Tras crear el estilo y mostrar la información sobre el rol del usuario y grupo, es necesario visualizar la imagen, consultando el manual de usuario de codeIgniter este nos facilita una librería llamada upload, la cual incluiremos en nuestra carpeta libreries, necesaria para poder subir archivos a nuestro proyecto en codeIgniter. Primero he creado una carpeta llamada imágenes donde se almacenarán las imágenes subidas en Asistencia->assets->images. He creardo un controlador imagen para que se encargara de subir las imágenes a esa carpeta además de guardar el nombre de esa imagen en la base de datos del usuario correspondiente. En este controlador he creado dos métodos:
    a. SubirImagen(): Se encarga de establecer la configuración con la que deseamos que se suban las imágenes (Ilustración 48) de cada usuario y de comprobar esta, cumple con los requisitos de formato y tamaño. Las imágenes se guardan en la carpeta

Asistencia->assets->Images del proyecto. En caso de que se produzca algún tipo de error por la imagen seleccionada, ya sea por tamaño o formato, se mostrará por pantalla.

```php
/**
* Comprueba la imagen que se va a subir por el usuario
* y se crea una miniatura para esa imagen
*
* @access public
* @param
* @return
*/

public function subirImagen(){
    $config['upload_path'] = realpath(APPPATH . '../assets/images');
    $config['allowed_types'] = 'gif|jpg|png';
    $config['max_size'] = '2048';
    $config['max_width'] = '1024';
    $config['max_height'] = '768';

    $this->load->library('upload',$config);
    $this->upload->initialize($config);

    if (!$this->upload->do_upload('fileImagen')) { // la imagen no cumple los requisitos
      $error = array('error' => $this->upload->display_errors());
      $this->load->view('layout/header');
      $this->load->view('perfil/vistaPerfil',$error);
      $this->load->view('layout/footer');

  } else {

      $file_data = $this->upload->data();
      $data = array('upload_data' => $this->upload->data());
      $this->crearMiniatura($file_data['file_name']);
      $imagen = $file_data['file_name'];
          $this->session->unset_userdata ( 'imagen' ); // borrammos la imagen anterior
          $this->session->set_userdata ( 'imagen' , $imagen );
          $this->usuario_model->cambiarImagen($imagen,$this->session->userdata('id'));
          //$this->session->set_userdata('imagen', $imagen);

          $data['error']="";
          $this->load->view('layout/header');
          redirect('inicio/vistaPerfil',$data);
          $this->load->view('layout/footer');
      }
  }
```

Ilustración 48. Función subir Imagen.

b. CrearMiniatura($fileimagen): Se encarga de subir la imagen seleccionada en la carpeta imágenes, además de asociar dicha imagen al usuario que la ha subido, mediante user_data(imagen).

Rule	Parameter	Description	Example
uploaded	Yes	Fails if the name of the parameter does not match the name of any uploaded files.	uploaded[field_name]
max_size	Yes	Fails if the uploaded file named in the parameter is larger than the second parameter in kilobytes (kb).	max_size[field_name,2048]
max_dims	Yes	Files if the maximum width and height of an uploaded image exceeds values. The first parameter is the field name. The second is the width, and the third is the height. Will also fail if the file cannot be determined to be an image.	max_dims[field_name,300,150]
mime_in	Yes	Fails if the file's mime type is not one listed in the parameter.	mime_in[field_name,image/png,image/jpg]
ext_in	Yes	Fails if the file's extension is not one listed in the parameter.	ext_in[field_name,png,jpg,gif]
is_image	Yes	Fails if the file cannot be determined to be an image based on the mime type.	is_image[field_name]

Ilustración 49. Reglas Subir Imagen.

Por defecto al crear un usuario le asignare al valor de su foto el nombre de una llamada estandar.jpg (Ilustración 50), para que cuando se cree un usuario, este por defecto tendrá asociada dicha imagen, de forma que al acceder por primera vez al perfil aparezca esta imagen y el usuario recuerde que debe cambiarla.

Ilustración 50. Imagen estandar.

Pasare a continuación, a crear la vista de crear usuario (Ilustración 51) y su funcionalidad. Como se ha especificado anteriormente, esta vista solo es accesible para los usuarios con rol de administrador y jefe de departamento. Esta opción será accesible desde el menú lateral de la página web. Una vez de pulsada la opción Crear Usuario, aparecerá por pantalla un formulario con los distintos campos a rellenar.

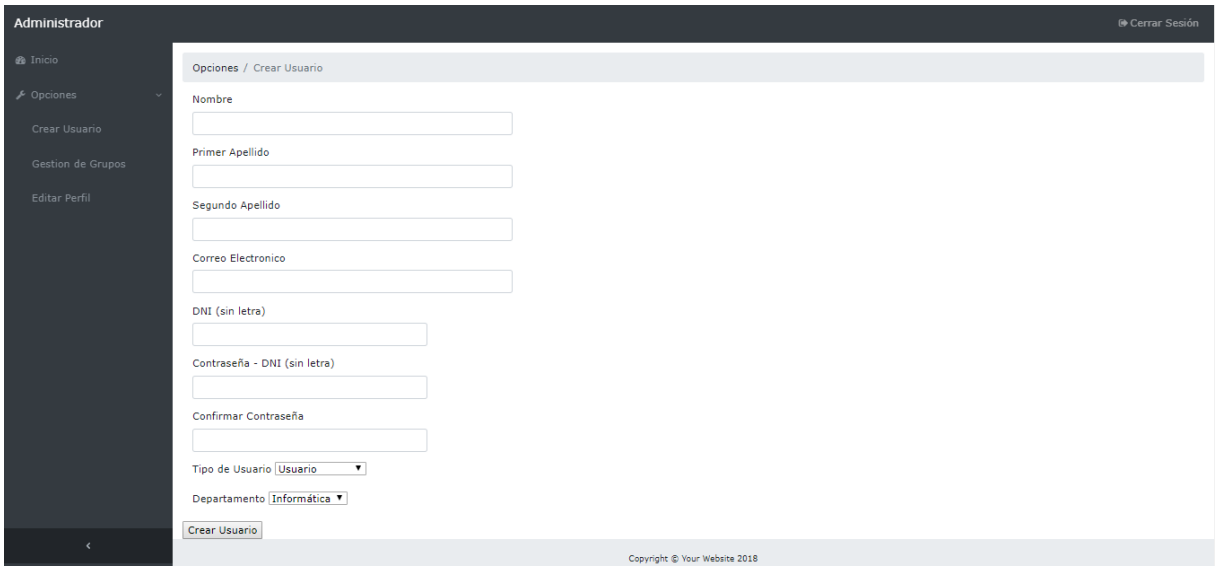


Ilustración 51. Vista Crear Usuario.

CodeIgniter proporciona una clase completa de preparación de datos y validación de formularios que ayuda a minimizar la cantidad de código. Utilizare esta validación para que una vez pulsado el botón crear usuario se compruebe que los campos introducidos son correctos, de lo contrario aparecerá por pantalla un mensaje indicando el error asociado al fallo o por el contrario un mensaje de confirmación de creación del usuario. También realizare dos comprobaciones en la base de datos, la primera es buscar que el DNI introducido no se encuentra ya en la base de datos lo que supondría que ese usuario ya está dado de alta. Por otra parte, también hare la misma comprobación con el email y asegurarse que no está siendo utilizado por otro usuario ya que supondría un problema a la hora de iniciar sesión puesto que tanto el DNI como el email deben ser únicos. En la Ilustración 52 puede verse el código asociado.

```php
/**
 * Obtiene el formulario de creacion de usuario y valida los credenciales,
 * si el formulario es correcto muestra un mensaje de exito u error en caso contrario
 *
 * @access public
 * @param
 * @return
 */

public function addUsuario(){
 $this->form_validation->set_rules('nombre','Nombre','max_length[50]|required');
 $this->form_validation->set_rules('primer_apellido','primer_apellido','max_length[50]|required');
 $this->form_validation->set_rules('segundo_apellido','segundo_apellido','max_length[50]|required');
 $this->form_validation->set_rules('email','Email','max_length[50]|required|valid_email');
 $this->form_validation->set_rules('dni','dni','exact_length[8]|required');
 $this->form_validation->set_rules('password','Password','min_length[6]|max_length[12]|required');
 $this->form_validation->set_rules('confirmPassword','ConfirmPassword', 'required|matches[password]');


 if ($this->form_validation->run($this)==FALSE){
  $data['grupos']=$this->grupo_model->getGrupos();
  $this->load->view('layout/header');
  $this->load->view('crearUsuario',$data);
  $this->load->view('layout/footer');

 }else{
   if($this->usuario_model->buscarUsuario($_POST['dni']) == 0 && $this->usuario_model->buscarEmail($_POST['email'])== 0 ){
    $hash = $this->bcrypt->hash_password($_POST['password']);
    $this->usuario_model->crearUsuario($_POST['nombre'],$_POST['primer_apellido'],$_POST['segundo_apellido'],$_POST['dni'],$_POST['grupo'],$_POST[
      'email'],$hash,$_POST['rol']);
    $data['grupos']=$this->grupo_model->getGrupos();
    $this->load->view('layout/header');
    echo "<script>alert('Usuario Creado');</script>";
    redirect('inicio/crearUsuario','refresh');
    $this->load->view('layout/footer');



   }else{
     $data['grupos']=$this->grupo_model->getGrupos();
     $this->load->view('layout/header');
     echo "<script>alert('Usuario con DNi o email ya existen');</script>";
     redirect('inicio/crearUsuario', 'refresh');
     $this->load->view('layout/footer');

 }
}
```

Ilustración 52. Validación crear usuario.

He creado dos desplegables al final de la vista de crear usuario (Ilustración 53). En el primer desplegable aparecerán los distintos roles a los que se puede asociar al nuevo usuario, teniendo en cuenta que si el usuario que ha iniciado la sesión en la página web es un jefe de departamento solo aparecerá la opción de crear un usuario normal, por el contrario, si ha sido el administrador puede escoger entre usuario y jefe de departamento. En el otro desplegable aparecerá los distintos grupos dados de alta en la base de datos, y al igual que en el desplegable anterior, si es el administrador el que está creando al usuario, este podrá asignarlo a cualquier departamento, mientras que si es el jefe de departamento el que crea al usuario, solo podrá asociárselo a su mismo departamento.

```
<div class="col-md-3 mb-3">
  <label for="validationDefault03">Tipo de Usuario </label>
  <select name="rol">
    <?php if($this->session->userdata('rol') == '0' ) {?>
    <option value="0" > Administrador </option>
    <option value="1" > Jefe de departamento </option>
    <?php } ?>
    <option value="2" selected > Usuario </option>
  </select>
</div>
<div class="col-md-3 mb-3">
  <label for="validationDefault05">Departamento</label>
  <select name="grupo">
    <?php if($this->session->userdata('rol') == '0'){
      foreach  ($grupos->result()  as  $row) { ?>
      <option value="<?php echo $row->nombre_grupo; ?>" selected><?php echo $row->nombre_grupo; ?></option>
      <?php } }else { ?>
      <option value="<?php echo $this->session->userdata('grupo'); ?>" selected><?php echo $this->session->userdata('grupo'); ?></option>
      <?php } ?>
  </select>
```

Ilustración 53. Desplegables Vista crear usuario.

4.2.4. Pruebas

Para este incremento la fase de pruebas consistirá en crear un usuario de forma correcta. Como aún no se puede seleccionar entre los diferentes departamentos, aparecerá por defecto el departamento de informática.

El resultado es el que aparece en la Ilustración 54 y la Ilustración 55.

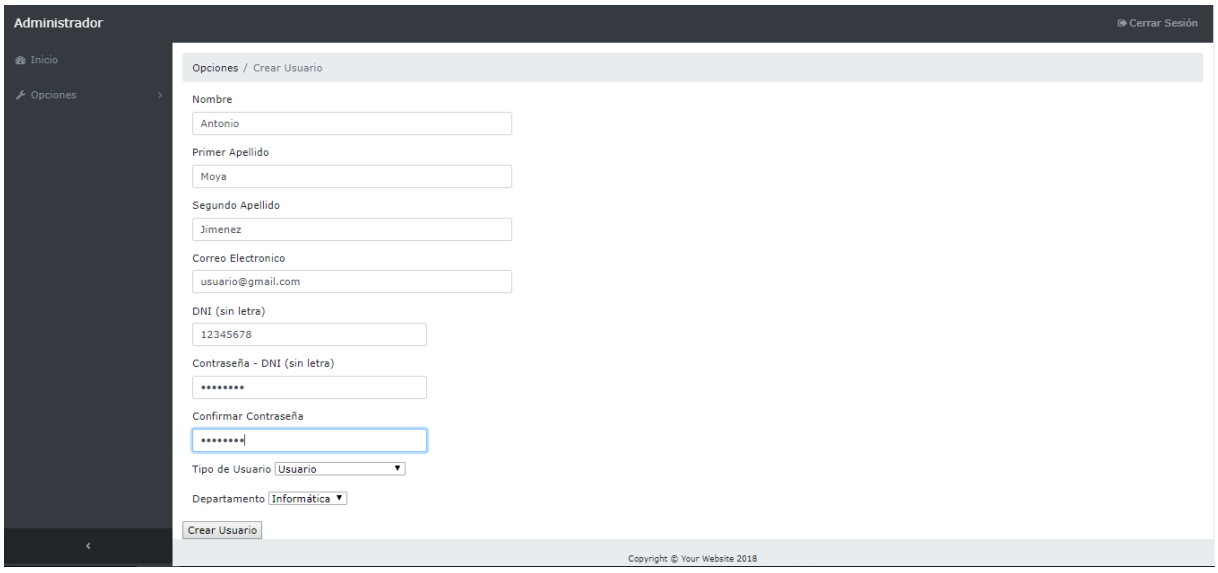


Ilustración 54. Pruebas 2 incremento -1.

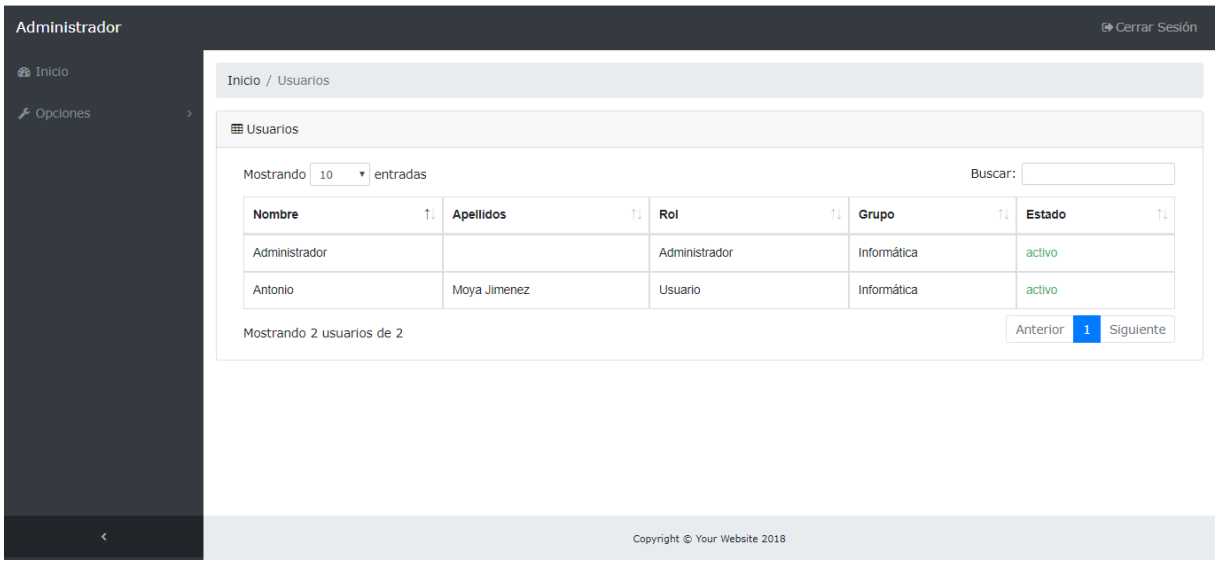


Ilustración 55. Pruebas 2 incremento -2.

4.3. Tercer Incremento

En este tercer incremento se procederá a la realización de la funcionalidad relacionada con la Gestión de Grupos, además de la creación de la vista principal de los usuarios con mayores privilegios del sistema.

4.3.1. Análisis

4.3.1.1. Casos de Uso

Como viene siendo habitual lo primero es crear el caso de uso asociado a este incremento. En este caso solo afecta al administrador ya que la tarea de gestión de grupo está asociado solo a usuarios con la mayor accesibilidad. Por lo tanto, el administrador tendrá las opciones de crear nuevos departamentos y asociarles un jefe de departamento y la de consultar estos grupos. En la Ilustración 56 puede verse su gráfico correspondiente.

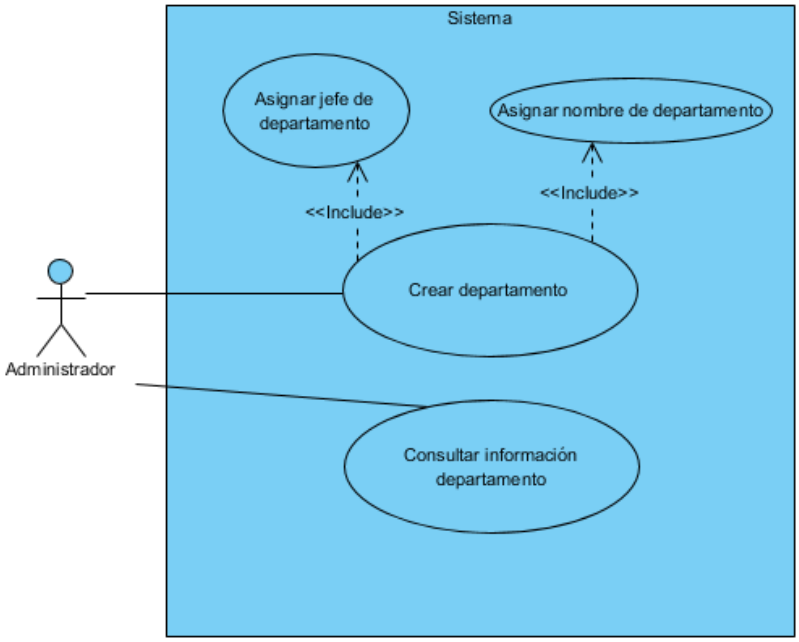


Ilustración 56. Gestión Departamentos.

4.3.2. Diseño

En lo relacionado al diseño de la interfaz de usuario, se crearán dos vistas diferentes, cada una con sus funcionalidades. La primera vista es la de la creación de nuevos departamentos por parte del administrador.

La segunda es la creación de las vistas principales tanto del administrador como de los jefes de departamento. En la que se creara una tabla con opciones de búsqueda y ordenación y la opción de poder acceder a los perfiles de los usuarios pulsando su fila en dicha tabla.

4.3.2.1. Diagrama de Secuencia

Para empezar, al igual que llevo haciendo con el resto de incrementos, creare el diagrama de secuencia asociado a la gestión de los departamentos, en la Ilustración 57 se puede observar el intercambio de mensajes en la línea temporal de los diferentes elementos que intervienen en la realización de la acción.

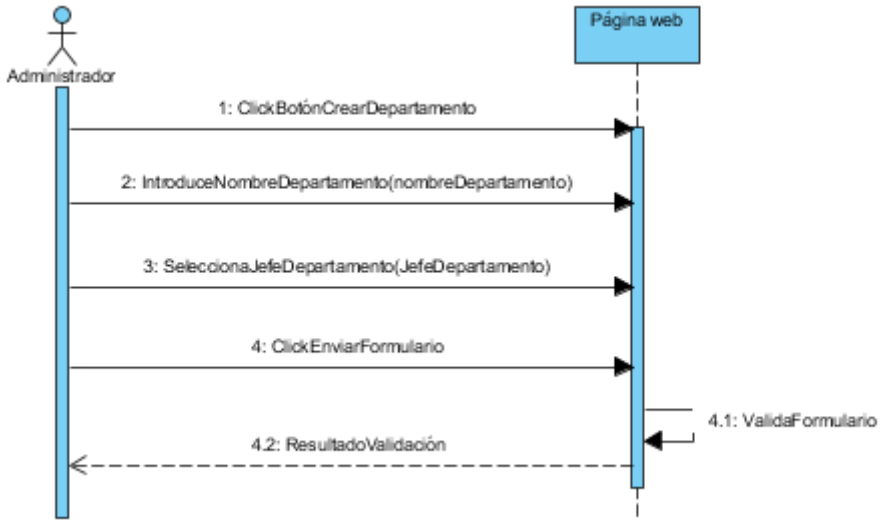


Ilustración 57. Diagrama de secuencia de crear departamento.

4.3.2.2. Interfaz de Usuario

En este aparatado se procederá a crear las vistas para la gestión de departamentos, donde aparecerán todos los departamentos y sus responsables y la vista de crear un nuevo departamento.

4.3.3. Implementación

Comenzare con la creación de la vista principal para los usuarios con rol jefe de grupo y administrador. Esta vista se compone básicamente de una tabla con la información a la que es accesible cada usuario, véase:

- Administrador: Tiene acceso a todos los usuarios dados de alta en el sistema.
- Jefe de Departamento: Tiene acceso a los usuarios asociados a su departamento, que hayan sido dados de alta por el mismo o por el administrador.

Con el objetivo de que sea más fácil la búsqueda de un usuario en particular, dicah tabla tendrá diferentes funcionalidades como:

- Búsqueda: Buscar un nombre concreto.

- Ordenación de la tabla por orden alfabético de nombre.
- Ordenación de la tabla por orden alfabético de apellidos.
- Ordenación por Tipo de usuario.
- Ordenación por Grupo.
- Ordenación por Estado del usuario (activo/inactivo).

De inicio en la tabla aparecerán las diez primeras entradas según el criterio de ordenación, pudiendo modificarse a 25, 50 y 100 en el caso de que en la aplicación existiera tal número de usuarios dados de alta en el sistema.

El código necesario para la obtención de los datos que aparecerán en la tabla (Ilustración 58), está en el controlador Inicio, donde básicamente están todas las funciones principales del sistema.

```
/**
 * Obtiene la tabla de usuarios relacionados con el usuario logueado,
 * Nota: para el superusuario obtiene todos los usuarios.
 * @access public
 * @param
 * @return
 */

public function obtenerTabla(){
 if($_SESSION['rol']==0){
  $grupos=$this->grupo_model->getGrupos();
  $usuariosFinales = array();
  foreach  ($grupos->result()  as  $grupo){
   foreach ($this->usuario_model->getUsuarios($grupo->nombre_grupo)->result()  as  $usuario){
    array_push($usuariosFinales,$usuario);
  }
}
$data = $usuariosFinales;
}else{
  if($_SESSION['rol']==1){
   $data = $this->usuario_model->getUsuarios($this->session->userdata('grupo'))->result();
 }else{
   $data = $this->usuario_model->getUsuario($this->session->userdata('id'));
 }
}
return $data;
}
```

Ilustración 58. Método obtener Tabla.

Una vez obtenida la información, esta se pasa a la vista tablaUsuarios, donde se van rellenando los distintos campos de la tabla (Ilustración 59).

```
<div class="card-body">
  <div class="table-responsive">
    <table class="sortable table table-hover" class= "table table-hover" id="dataTable" width="100%" cellspacing="0">
      <thead>
        <tr>
          <th>Nombre</th>
          <th>Apellidos</th>
          <th>Rol</th>
          <th>Grupo</th>
          <th>Estado</th>
        </tr>
        <tbody>
          <?php foreach  ($usuario  as  $row) {?>
          <tr class='clickable-row' data-id='<?=$row->id?>'>
            <?php if($row->rol == 0){
              $nombreRol='Administrador';
            }else if($row->rol == 1){
              $nombreRol='Jefe Departamento';
            }else{

              $nombreRol='Usuario';
            }
            ?>
            <td><?php echo $row->nombre; ?></a></td>
            <td><?php echo $row->primer_apellido." ". $row->segundo_apellido; ?></td>
            <td><?php echo $nombreRol; ?></td>
            <td><?php echo $row->grupo; ?></td>
            <?php if($row->estado == TRUE){?>
            <td style="color:MediumSeaGreen;"> <?php echo 'activo';
          }else{ ?>
          <td style="color:Tomato;"> <?php echo 'inactivo'; }?></td>
        </tr>

        <?php } ?>
      </tbody>
    </thead>
  </table>
</div>
```

Ilustración 59. Html tabla principal.

El resultado obtenido es el que aparece en la Ilustración 60.

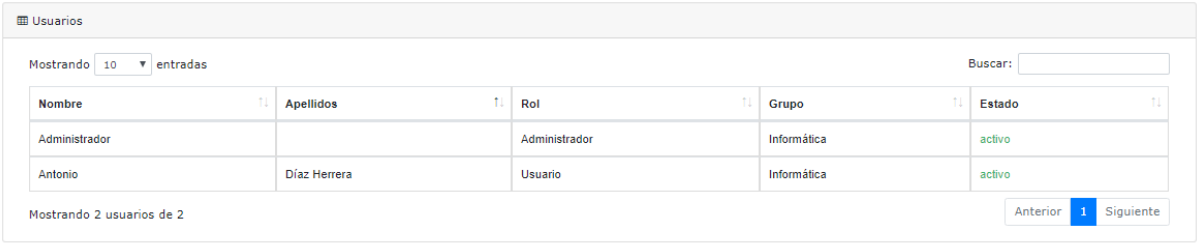


Ilustración 60. Resultado Tabla usuarios.

Pasare ahora a la implementación de toda lo relacionado con la gestión de grupos. Lo primero será crear nuevos archivos de tipos controlador y modelo que se encargaran de obtener y gestionar la información de los distintos departamentos. Serán los archivos grupo_model y gupo_controller. También he creado una carpeta

en vistas llamada grupo donde iré almacenando las vistas que cree durante esta implementación.

Es hora de pasar a la implementación de la vista Grupo. El propósito de esta vista es visualizar en una tabla todos los grupos a los que está asociado un jefe de grupo o en el caso de acceder como administrador información sobre todos los grupos creados en la base de datos. El acceso a esta vista se hará desde el menú lateral, clicando en Opciones->Gestión de Grupos y en el, mostrare una tabla con dos columnas:

- Nombre del Grupo: Con el nombre de los diferentes grupos.
- Responsable del grupo: Con el nombre del responsable de cada grupo.

Esta información la obtengo a través del controlador Grupo_controller a partir de un método llamado getGrupo, que devuelve las instancias que existen en la tabla grupo y accediendo a través del nombre de su variable accede al dato, como puede ser grupo->nombre_grupo. El controlador manda a la vista esta información y allí compruebo el rol del usuario logueado, si el rol es 1 significa que este usuario solo puede acceder a la información de sus grupos, si es 2 se muestra toda la información relacionada con los grupos. Además, he incluido un botón debajo de la tabla para crear nuevos grupos cuya implementación se explicará a continuación. En la Ilustración 61 puede verse el resultado final de esta vista.

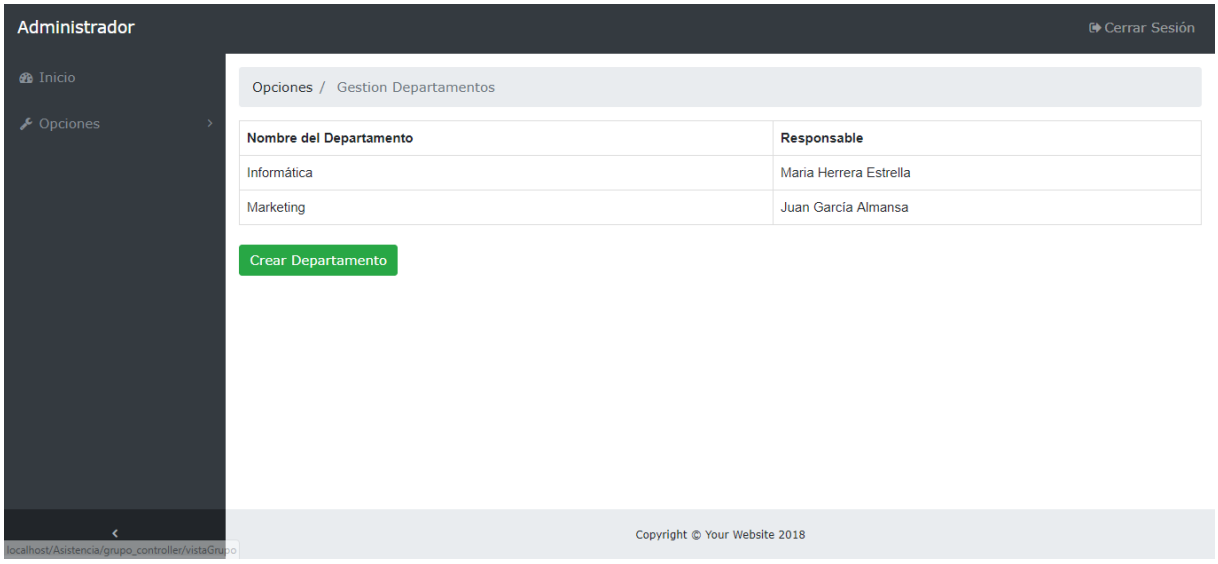


Ilustración 61. Resultado Gestión Grupos.

También creare la vista crearGrupo en la que básicamente aparece un formulario con dos campos, uno para asignar el nombre del grupo a crear y otro para seleccionar el responsable del grupo. En este último campo se selecciona entre un conjunto de opciones que depende del usuario que acceda a la página, si el rol del usuario es 1, no se podrá seleccionar ningún responsable salvo el que ha iniciado sesión, si el rol es 2 se mostraran todos los usuarios con rol igual a 2 obtenidos a partir del método getUsuario() del modelo grupo_model. Dispone de un botón al final del formulario para enviarlo al controlador grupo_controller y que este realice las comprobaciones que he establecido (Ilustración 62), como son:

- Que el campo de nombre de grupo no este vacío ni tenga más de 50 caracteres.
- Que el nombre introducido no exista ya en la base de datos.

Si las comprobaciones tienen éxito el controlador llama al método nuevoGrupo del modelo Grupo_model con la información facilitada y crea una nueva instancia en la base de datos, y es entonces cuando el controlador le facilita a la vista un mensaje con el resultado de la operación, diciendo si ha tenido éxito o no.

```php
/**
*  Recibe un formulario con el nombre del grupo y comprueba si existe
*
*
* @access public
* @param
* @return
*/

public function crearGrupo(){
   if($this->input->post('submit')){
        //comprobamos que no exista ese grupo
       if($this->grupo_model->buscarGrupo($_POST['nombre_grupo'])==0){
           $this->grupo_model->crearGrupo($_POST['nombre_grupo'],$_POST['id_usuario']);

           $data['usuarios']= $this->usuario_model->getUsuariosRolDistinto('1')->result();
           $this->load->view('layout/header');
           echo "<script>alert('Grupo creado con exito');</script>";
           redirect('grupo_controller/vistaCrearGrupo','refresh');
           $this->load->view('layout/footer');

       }else{

           $data['usuarios']= $this->usuario_model->getUsuariosRolDistinto('1')->result();
           $this->load->view('layout/header');
           echo "<script>alert('Este grupo ya existe');</script>";
           redirect('grupo_controller/vistaCrearGrupo','refresh');
           $this->load->view('layout/footer');

       }

   }
}


}
```

Ilustración 62. Método crear grupo.

El resultado de la vista puede verse en la Ilustración 63.

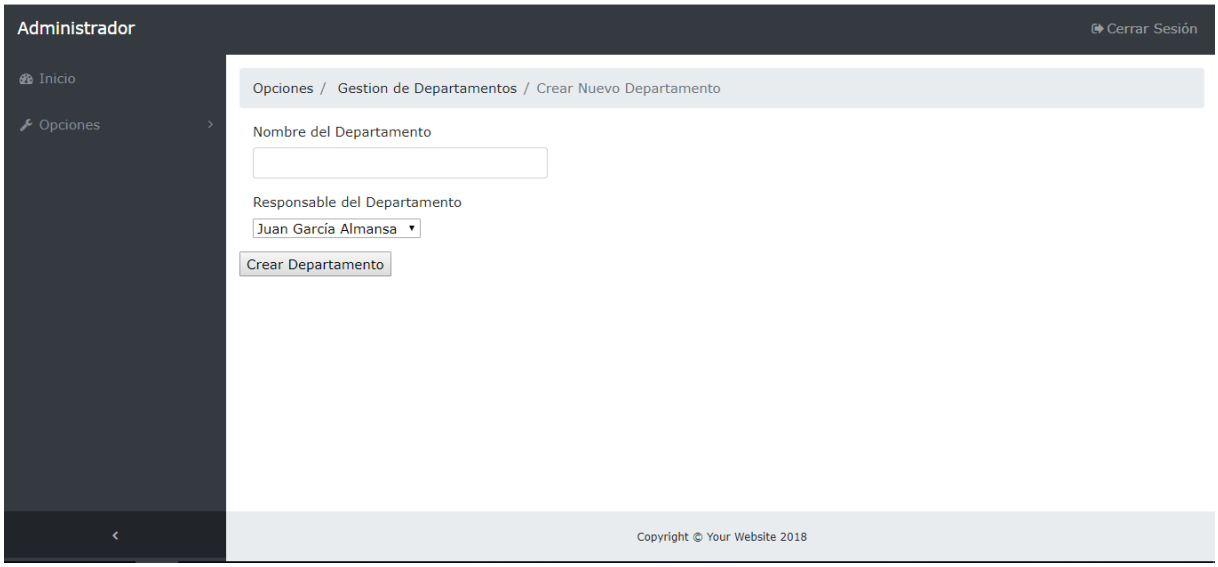


Ilustración 63. Resultado Crear Departamento.

Una vez creadas las vistas pasaré a organizar su visualización como está indicado en el diseño. Para empezar trabajare con el controlador Inicio, en el método index que se encarga del login de la página web, tras realizar la comprobación de los credenciales y crear el user_data(usuario), comprobamos que tipo de usuario es el que se ha logueado, si el rol = 2 se muestra la vista horario, si rol=1 o rol=0 se muestra la vista tablaUsuario.

4.3.4. Pruebas

A continuación, se llevará a cabo la fase de pruebas de este incremento, que consistirá en crear un nuevo grupo y asociarlo a un jefe de departamento. Para ello, primero hay que crear un nuevo usuario con el rol de jefe de departamento, en este caso será “Juan García Almansa” y después, crear un nuevo departamento que se llamará “Marketing” y asociárselo a dicho usuario. Los resultados son los mostrados en la Ilustración 64y la Ilustración 65.

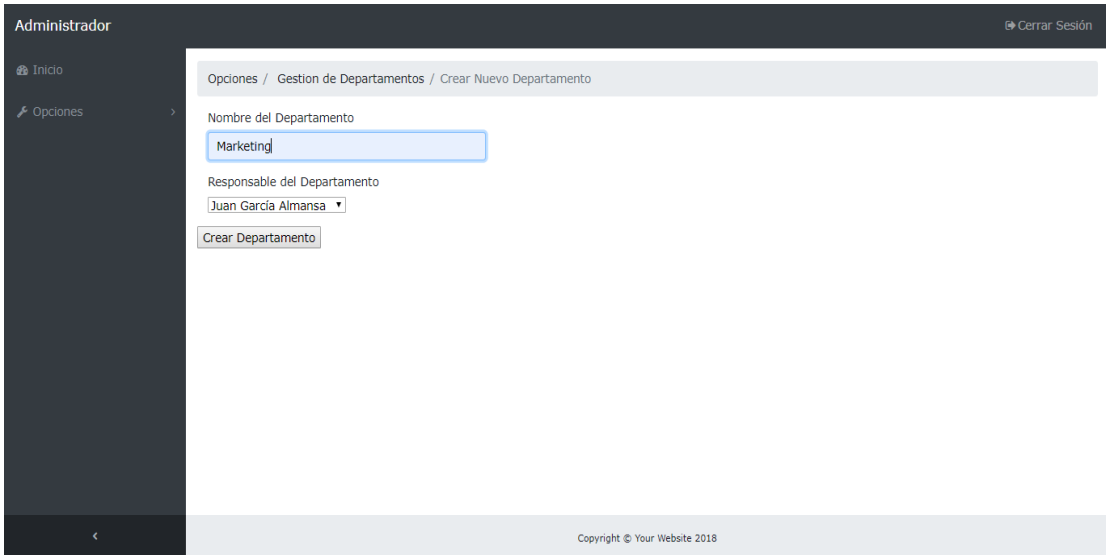


Ilustración 64. Creación nuevo Departamento.

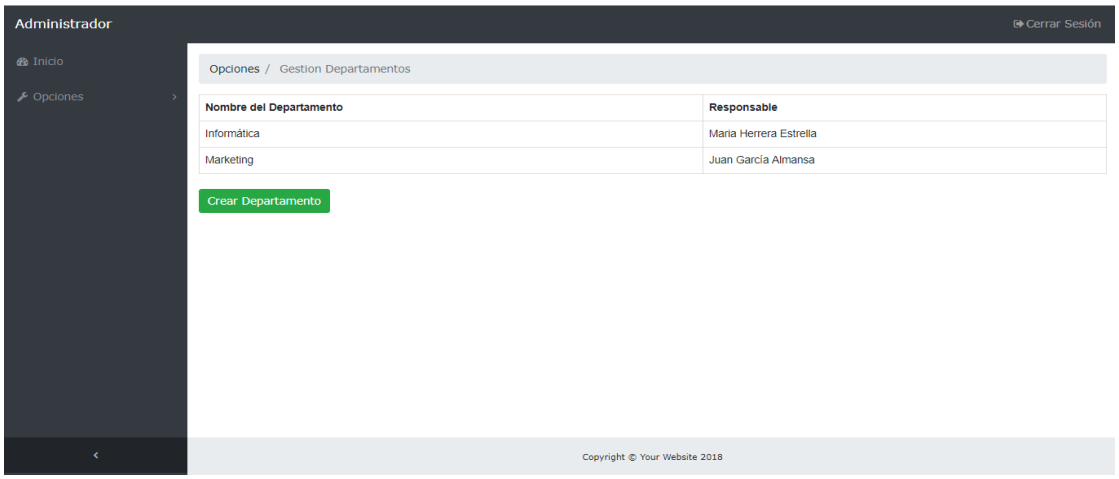


Ilustración 65. Resultado crear nuevo departamento.

4.4. Cuarto Incremento

Ha llegado el momento de comenzar con el desarrollo del módulo de identificación, que se encargará de enviar al servidor la información sobre la hora a la que un usuario se ha identificado en el sistema y almacenarlo en la base de datos. Este incremento es junto al primero, uno de los más largos de desarrollar, dado que es necesario implementar simultáneamente cosas relacionadas con el módulo y con la página web.

4.4.1. Análisis

4.4.1.1 Casos de Uso

Empezaré con el análisis con todo lo que respecta al módulo de identificación. Para ello he creado un caso de uso que se puede apreciar en la Ilustración 66, en el que se muestra la interacción del usuario con el sistema y las diferentes acciones que puede realizar sobre él.

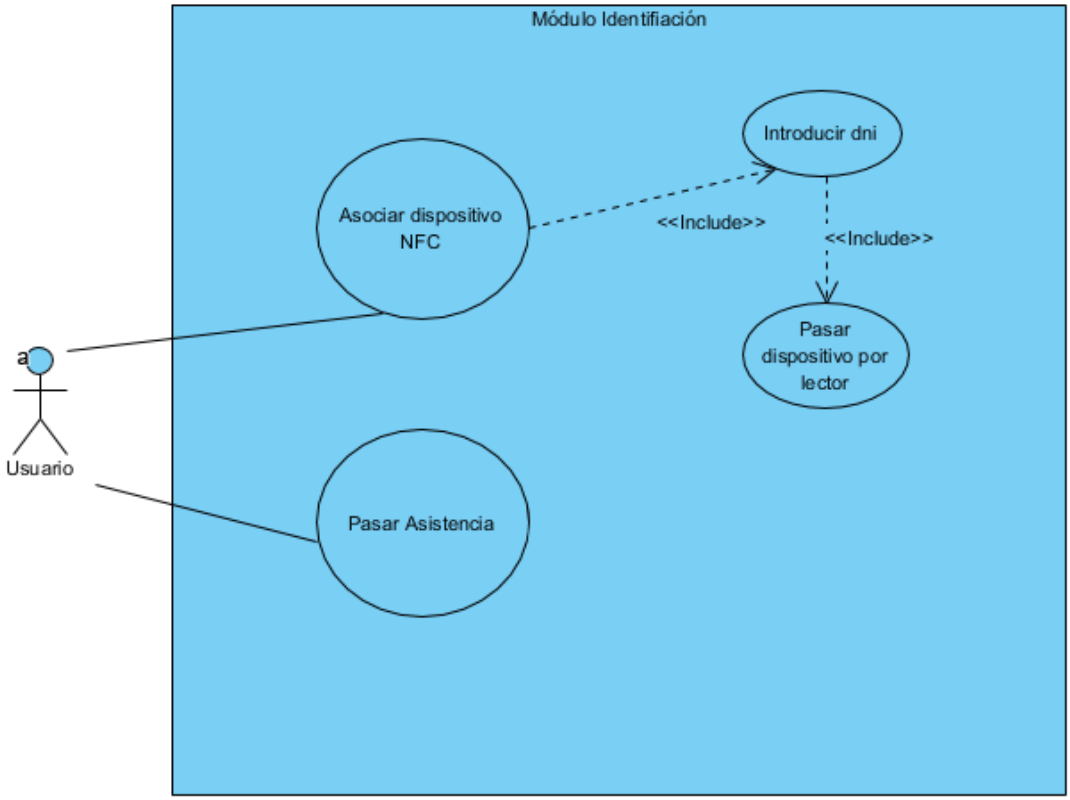


Ilustración 66. Caso de uso módulo de identificación.

Como se puede apreciar en la imagen anterior, es el propio usuario el que debe asociar el dispositivo NFC. Para ello primero introducirá su DNI en el módulo de identificación que dispondrá de un teclado, y después, pasará sobre el lector la tarjeta o llavero que desee asociar. Una vez asociada la tarjeta podrá pasar asistencia pasando la misma por el sistema de identificación.

4.4.1.2. Máquina de Estados

En cuanto al comportamiento del módulo de identificación, este será similar al de una máquina de estados por lo que seguirá el patrón que he creado en la Ilustración 67.

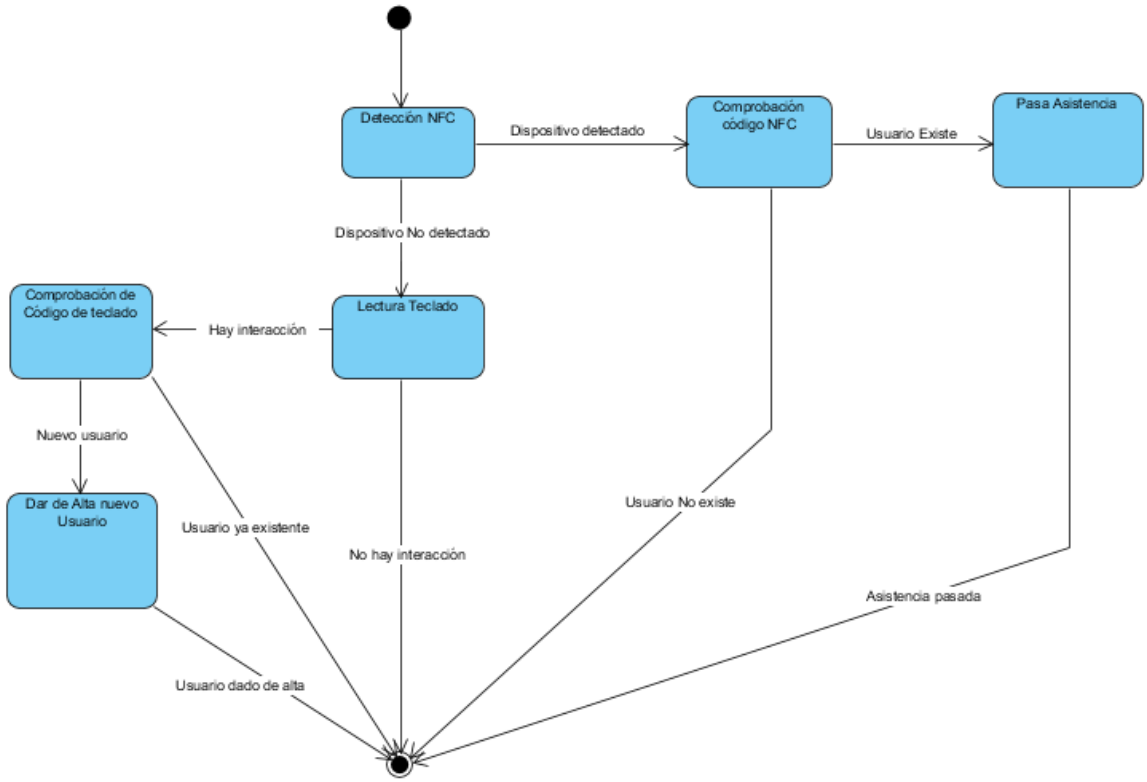


Ilustración 67. Máquina de Estados.

Según las respuestas recibidas en cada estado, se realizará una parte del código u otra. Hay que tener en cuenta que una vez llegado al estado final, el programa vuelve a repetirse.

4.4.2. Diseño

4.4.2.1. Diagrama de Secuencia

Como viene siendo común, al diagrama de casos de uso expuesto en el apartado anterior, le corresponde sus diagramas de secuencias donde se muestran el intercambio de mensajes con el sistema a través de la línea temporal. La Ilustración 68 para asociar un nuevo dispositivo, y la Ilustración 69 para pasar asistencia.

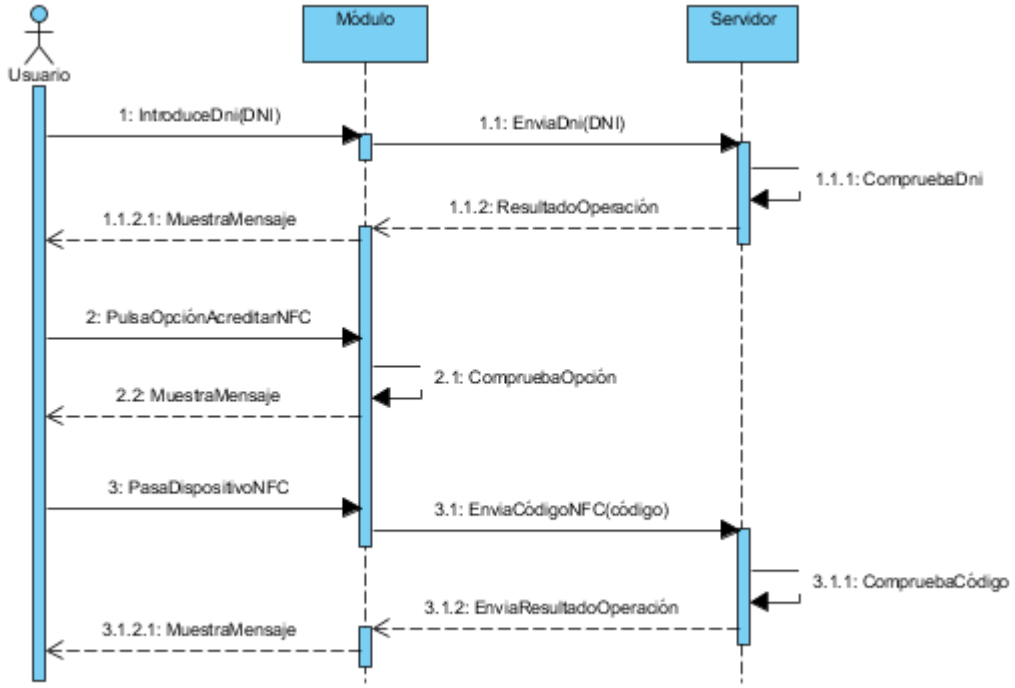


Ilustración 68. Diagrama de secuencia de asignar dispositivo NFC.

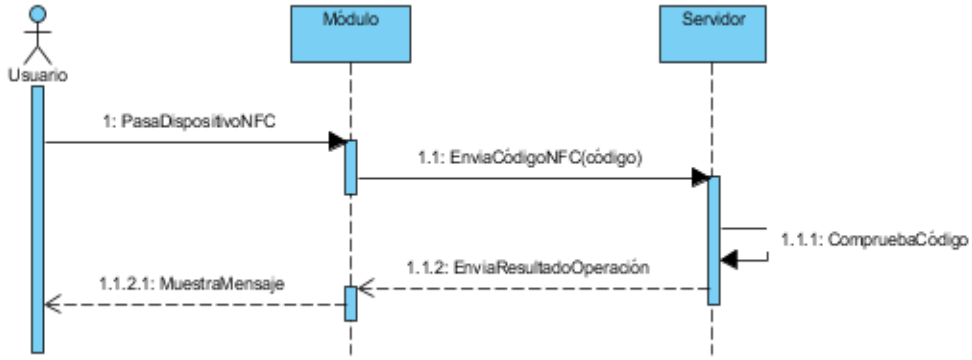


Ilustración 69. Diagrama de secuencia pasar asistencia.

Para el diseño del módulo de identificación, me concentraré en crear un prototipo capaz de interactuar con el usuario.

4.4.2.2. Interfaz de Usuario

En lo que respecta a la interfaz de usuario de la página web, la única vista que queda por implementar es la del horario de cada usuario donde debe aparecer una

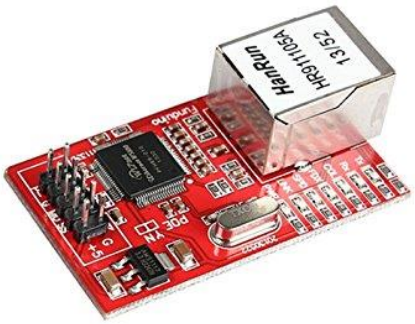
tabla con las horas de cada usuario a lo largo del mes y la opción de poner filtrar las horas por fecha.

4.4.4. Implementación

4.4.4.1. Módulo NodeMcu

Empezaré esta implementación con la creación del módulo de identificación. Se necesitan varios componentes que añadir a la placa NodeMcu, cada una de ellos necesita una librería específica para llevar a cabo su funcionamiento. En la Tabla 7 se muestra una imagen con el componente que he utilizado, junto con una breve descripción.

Componente	Descripción
	Este componente se encarga de convertir las señales eléctricas de la placa en información visual, de tal forma que es el componente encargado de mostrar mensajes al usuario cada vez que pasa la tarjeta NFC por el sensor. funciona a 5 voltios.
	El PN532 utiliza un sistema avanzado de modulación y demodulación para todo tipo de dispositivos pasivos de 13.56Mhz. También puesto que se hará una lectura y escritura de la tarjeta.

	El ENC28J60 es un controlador de Ethernet diseñado para sistemas embebidos. Se controla a través de bus SPI, por lo que la conexión con NodeMcu es muy sencilla. Opera a 3.3, pero es tolerante a señales de 5V, por lo que su integración es aún más sencilla.
	Está diseñado como un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro; el esclavo reacciona. Es posible tener varios maestros mediante un modo multimaestro, en el que se pueden comunicar dos maestros entre sí, de modo que uno de ellos trabaja como esclavo. El arbitraje (control de acceso en el bus) se rige por las especificaciones, de este modo los maestros pueden ir turnándose.
	Se utilizan de forma general para transferir señales eléctricas de cualquier parte de la placa de prototipos a los pines de entrada/salida de un microcontrolador.
	Un teclado matricial es un dispositivo que agrupa varios pulsadores y permite controlarlos empleando un número de conductores inferior al que necesitaríamos al usarlos de forma individual. Podemos emplear estos teclados como un controlador para un autómata o un procesador como NodeMcu.

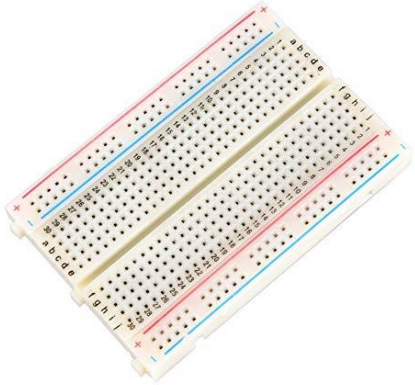
	La Protoboard, es una placa de pruebas en los que se pueden insertar elementos electrónicos y cables con los que se arman circuitos sin la necesidad de soldar ninguno de los componentes.

Tabla 7. Componentes

Antes de comenzar con la implementación, es necesario hacer un inciso y explicar cómo funcionan las comunicaciones con NodeMcu. Básicamente se pueden realizar las comunicaciones entre NodeMcu y distintos dispositivos de tres formas diferentes: a través del puerto serie, a través del bus SPI o utilizando el bus I2C.

- Puerto Serie: Los puertos serie son la forma principal de comunicar una placa NodeMcu con un ordenador. Gracias al puerto serie podemos, por ejemplo, mover el ratón o simular la escritura de un usuario en el teclado, enviar correos con alertas… Existen un sin fin de posibilidades en las que se requiere el empleo del puerto serie. Por tanto, el puerto serie es un componente fundamental de una gran cantidad de proyectos de NodeMcu, y es uno de los elementos básicos. Envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). Con el node MCU ESP… los pines empleados son 0 (RX) y 1 (TX). Básicamente hare uso de este tipo de comunicación para enlazar el node MCU al ordenador y poder enviar información utilizando el IDE NodeMcu (Ilustración 70).

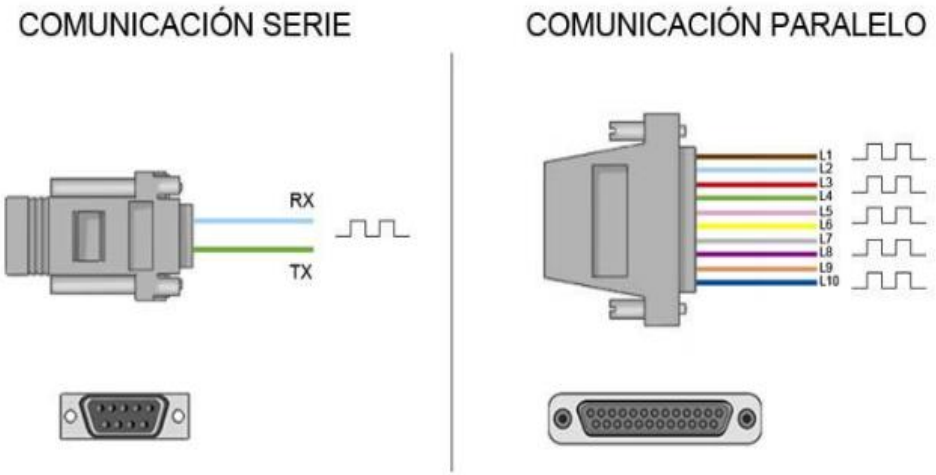


Ilustración 70. Comunicación Serie y Paralelo.

- Bus SPI: tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro (master) puede iniciar la comunicación con uno o varios dispositivos esclavos (slave), y enviar o recibir datos de ellos. Los dispositivos esclavos no pueden iniciar la comunicación, ni intercambiar datos entre ellos directamente. En el bus SPI la comunicación de datos entre maestros y esclavo se realiza en dos líneas independientes, una del maestro a los esclavos, y otra de los esclavos al maestro (Ilustración 71).

Ilustración 71. Bus Spi.

  - MOSI (Master-out, slave-in) para la comunicación del maestro al esclavo.
  - MISO (Master-in, slave-out) para comunicación del esclavo al maestro.
  - SCK (Clock) señal de reloj enviada por el maestro.
  - SS (Slave Select): para seleccionar el dispositivo con el que se va a realizar la comunicación.

Sin embargo, esto tiene la desventaja de requerir una línea por cada dispositivo esclavo. En caso de disponer muchos dispositivos esclavos esto puede no ser práctico, por lo que es posible adoptar una conexión en cascada, donde cada esclavo trasmite datos al siguiente, como se puede apreciar en la Ilustración 72.

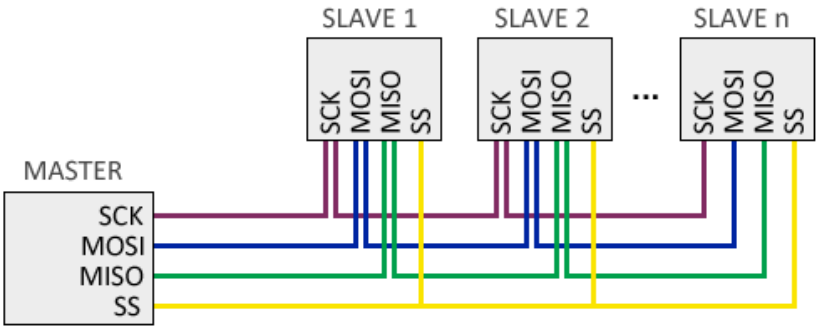


Ilustración 72. Conexión en cascada.

- Bus I2C: requiere únicamente dos cables para su funcionamiento, uno para la señal de reloj (CLK) y otro para el envío de datos (SDA), lo cual es una ventaja frente al bus SPI, ya que nos ahorramos pines que necesitaremos para las conexiones posteriores. En el bus Cada dispositivo dispone de una dirección, que se emplea para acceder a los dispositivos de forma individual. Tiene una arquitectura de tipo maestro-esclavo, al igual que el bus SPI. El bus I2C es síncrono. El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. De esta forma, se elimina la necesidad de que cada dispositivo tenga su propio reloj (Ilustración 73).

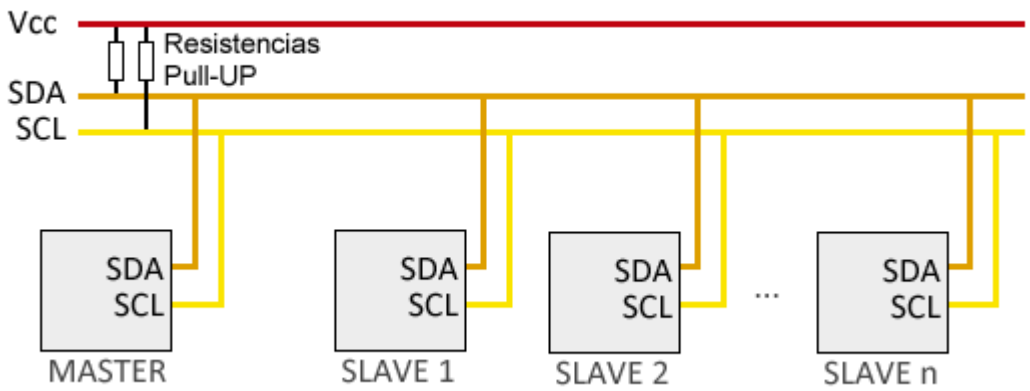


Ilustración 73. Ejemplo Bus I2C.

En la Tabla 8 se muestra una pequeña comparación de las características del bus Spi y el bus I2c.

Bus Spi	Bus I2c
Es full duplex	No es full duplex
Alta velocidad de transmisión	Su velocidad es media-baja
Se requiere 3 cables (SCK, MOSI y MISO) + 1 cable adicional (SS) por cada dispositivo esclavo.	Requiere pocos cables
No se dispone de ningún mecanismo de control, es decir, no podemos saber si el mensaje ha sido recibido y menos si ha sido recibido correctamente.	Dispone de mecanismos para verificar que la señal ha llegado

Tabla 8. Comparación Bus Spi y bus I2c.

Empezare con la comunicación entre el lcd y NodeMcu. La comunicación se realizará a través del bus I2c, para ello he incluido en el IDE de NodeMcu la librería “LiquidCrystal_I2C.h” para así crear un objeto led. La librería dispone de cuatro métodos:

- Lcd.begin(): Para inicializar el led.
- Lcd.set(pos_fila,pos_columna): Para indicar la posición donde desea mostrar texto.

- Lcd.print(“texto”): Para imprimir texto en el led.
- Lcd.clear(): Para borrar el texto escrito en el led.

El siguiente componente a conectar es el teclado 3x4 por el que los usuarios interactuaran para darse de alta en el sistema con NodeMcu. Al igual que el lcd utilizara el bus i2c por lo que he incluido la librería “Keypad_i2c”, tras crear un objeto keypad (kpd), hago uso de los siguientes métodos:

- Kpd.begin(): Para inicializar el teclado.
- Kpd.get_key():Para obtener la tela pulsada.

Es el turno del módulo ethernet necesario para obtener conexión a internet y así poder empezar a crear la máquina de estados de NodeMcu. La comunicación en este caso se realizarla mediante el bus Spi. He incluido la librería “ethernet.h” necesaria para realizar la conexión a la red a través de un cable ethernet.

El módulo NFC utiliza los pines TX y RX necesarios para la comunicación en serie entre NodeMcu y el ordenador. Por lo que comprobare de forma individual el correcto funcionamiento de este módulo utilizando los pines que dejan libres los otros dispositivos. Una vez conectado el lector utilizare las librerías “PN532_HSU.h” y “PN532.h” para inicializar el objeto NFC. El correcto funcionamiento de este módulo ha sido sin duda el más complejo del sistema debido a que muchas de las librerías que he encontrado no eran compatibles con ese modelo. Además, hay que sumarle que necesitaba que además de funcionar con los pines digitales que nos ofrece NodeMcu, también funcionara cuando solo disponía de los pines del serial. Tras probar distintas configuraciones, obtuve el resultado esperado, mostrar por el monitor el código de asociado a los dispositivos NFC, solo son válidos aquellos que tienen una longitud de 4-7 bytes, por lo que algunas tarjetas no son compatibles con este módulo.

He de añadir que tuve la idea de que el Código NFC asignado pudiera asociarse a un dispositivo móvil, ya que la mayoría disponen de esta tecnología, el problema fue que al detectar varias veces el código a través del NodeMcu con un

mismo dispositivo móvil, este en cada una de las lecturas, proporcionaba un código diferente, por lo que tuve que desechar esa idea.

Tras comprobar el correcto funcionamiento del módulo NFC, pasare a crear las partes de la máquina de estados, salvo que no utilizare el lector dado que necesito esos pines para la comunicación por serie, por lo que usare un código aleatorio simulando el código que proporcionaría el lector NFC.

Empezare con la parte de asignar una tarjeta NFC, como se muestra en la Ilustración 74.

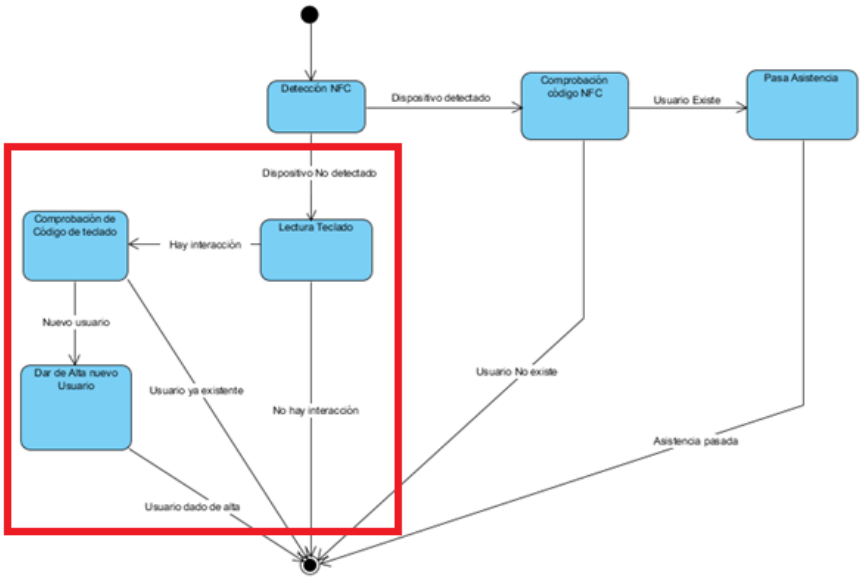


Ilustración 74. Primera implementación máquina de estados.

Para entrar en esta fase de la máquina de estados, es necesario pulsar el teclado de NodeMcu, y teclear un código de 8 dígitos, correspondiente al DNI del usuario que quiere activar su tarjeta NFC. Tras recibir los 8 dígitos, se crea un mensaje HTTP que enviara el mensaje al servidor, que comprobara si existe en la base de datos. Tras la comprobación, enviará un mensaje con la respuesta de vuelta a NodeMcu, que gracias a la librería “NodeMcuJson.h” obtendrá la información necesaria. Todo este proceso es el que aparece en la Ilustración 75.

```
if (posicion == 8 && !nuevo) { // comprobamos que se han introducido los 8 correctamente (DNI)
  // wait for WiFi connection
  String codigo;
  for (int i = 0; i < 8; i++) {
    codigo += cod[i];
  }
  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    //Serial.println("Failed to configure Ethernet using DHCP");
    // try to congifure using IP address instead of DHCP:
    Ethernet.begin(mac, ip);
  }
  // give the Ethernet shield a second to initialize:
  delay(1000);
  //Serial.println("connecting...");

  // if you get a connection, report back via serial:
  if (client.connect(ip, 8033)) {
    lcd.println("connected");
    // Make a HTTP request:
    client.println("GET /Asistencia/Api_controller/codigo?codigo=" + codigo + "HTTP/1.1");
    client.println("Host: 150.214.174.25:8033");
    client.println("User-Agent: Arduino-Ethernet");
    client.println("Connection: close");
    client.println("Content-Type: application/x-www-form-urlencoded;");
    client.println();

    char status[32] = {0};
    client.readBytesUntil('\r', status, sizeof(status));
    // Skip HTTP headers
    char endOfHeaders[] = "\r\n\r\n";
    if (!client.find(endOfHeaders)) {
      //Serial.println(F("Invalid response"));
      return;
    }

    // Allocate JsonBuffer
    // Use arduinojson.org/assistant to compute the capacity.
    size_t capacity = JSON_OBJECT_SIZE(3) + JSON_ARRAY_SIZE(2) + 60;
    DynamicJsonBuffer jsonBuffer(capacity);

    // Parse JSON object
    JsonObject& root = jsonBuffer.parseObject(client);
    if (!root.success()) {
      //Serial.println(F("Parsing failed!"));
      return;
    }
```

Ilustración 75. Código teclado.

A partir de aquí pueden ocurrir dos cosas; si la respuesta es “registrado”, significa que el usuario ya tiene asignado una tarjeta NFC, por lo que saldremos de ese estado y mostrara por el lcd de NodeMcu “Usuario ya asignado”, por otra parte si la respuesta es “nuevo”, saldremos de ese estado para entrar en otro nuevo de la máquina de estados, la de asignar el código NFC a un usuario, es decir, llamara al método asignar() el cuál aparece en la Ilustración 76.

```
void msgOpcion() {
  lcd.print("Asignar NFC");
  lcd.setCursor(2, 1);
  lcd.print("Si: *");
  lcd.setCursor(8, 1);
  lcd.print("No: #");
}

int asignar() {
  char opt = kpd.getKey() ;
  if (opt == '*') {
    return 1;
  } else if (opt == '#') {
    return 0;
  }
  return -1;
}
```

Ilustración 76. Método Asignar.

En este estado se mostrará por pantalla el siguiente texto “Asignar NFC * Si # No” NodeMcu esperara la respuesta a través del teclado, si se pulsa “*”, NodeMcu esperará la información asociada a la tarjeta NFC, más adelante se introducirá el código obtenido por el módulo NFC. Si se pulsa la tecla “#” saldremos de ese estado y no se almacenará ninguna información.

La última parte de la máquina de estados es la lectura de dispositivo NFC como se ve en la Ilustración 77.

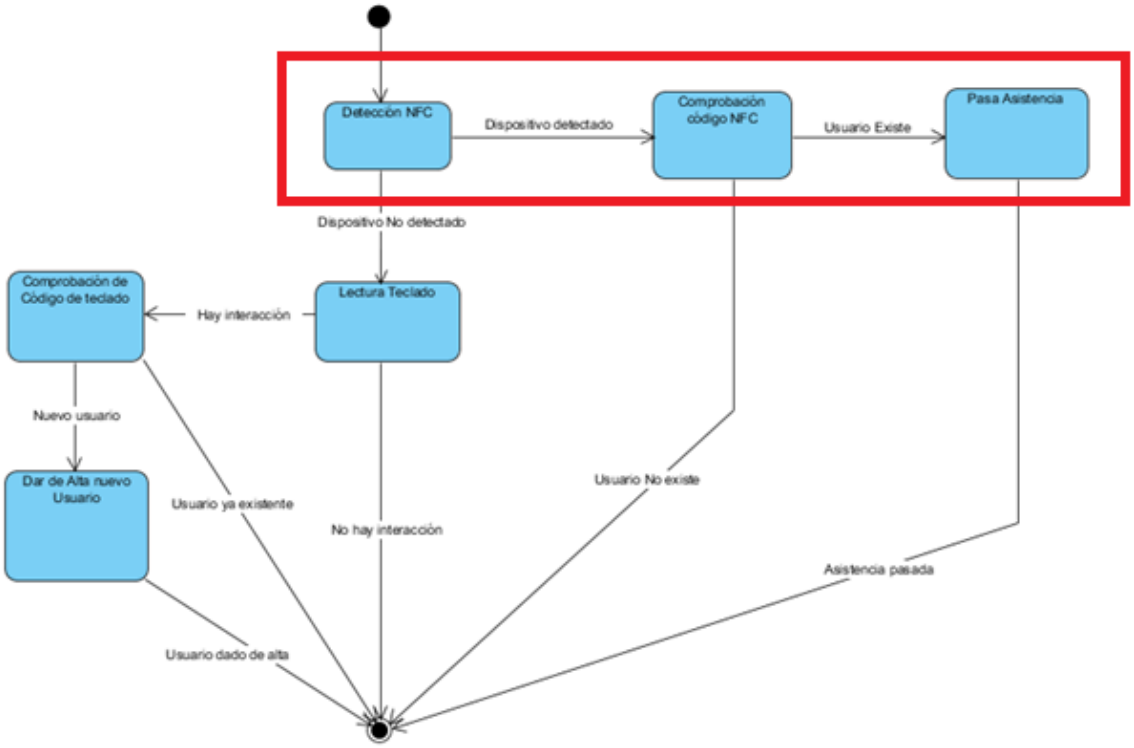


Ilustración 77. Segunda pate máquina de estados.

Esta parte se activará si se detecta una lectura de algún dispositivo NFC. He modificado el tiempo de lectura de los dispositivos NFC que nos facilitaba la librería “PN532.h”, para que tarde al menos 1 segundo en volver a comprobar una nueva lectura, ya que, de lo contrario, no dejaba a NodeMcu evaluar los otras posibilidades de la máquina de estados. Esta es quizás la parte más simple de la máquina de estados ya que tras detectar y guardar el código de la tarjeta, se creará un mensaje HTTP y se enviara el código detectado al servidor como se aprecia en la Ilustración 78.

```
//NFC////////////////////////////////////////////////////////////////
if (!teclado) {
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength, 100);// leemos tarjeta (tiempo de lectura alterado)
  if (success) {
    String result;
    for (byte i = 0; i < uidLength; i++) {
      if (uid[i] < 10) {
        result += "0";
      }
      result += String(uid[i], HEX);
    }

    if (Ethernet.begin(mac) == 0) {
      //Serial.println("Failed to configure Ethernet using DHCP");
      // try to congifure using IP address instead of DHCP:
      lcd.clear();
      lcd.print("Fallo conexion");
      Ethernet.begin(mac, ip);
    }
    // give the Ethernet shield a second to initialize:
    delay(1000);
    //Serial.println("connecting...");

    // if you get a connection, report back via serial:
    if (client.connect(ip, 8033)) {
      //Serial.println("connected");
      // Make a HTTP request:
      client.println("GET /Asistencia/Api_controller/tarjeta?tarjeta=" + result + " HTTP/1.1");
      client.println("Host: 150.214.174.25:8033");
      client.println("User-Agent: Arduino-Ethernet");
      client.println("Connection: close");
      client.println("Content-Type: application/x-www-form-urlencoded;");
      client.println();

      char status[32] = {0};
      client.readBytesUntil('\r', status, sizeof(status));
      // Skip HTTP headers
      char endOfHeaders[] = "\r\n\r\n";
      if (!client.find(endOfHeaders)) {
        //Serial.println(F("Invalid response"));
        return;
      }
```

Ilustración 78. Código Lectura NFC.

Tras realizar las comprobaciones oportunas, NodeMcu obtendrá una respuesta Json con tres posibles resultados (Ilustración 79):

- Nombre de usuario, entrada: Mostrará por pantalla “Bienvenido (nombre usuario)”.
- Nombre de usuario, salida: Mostrará por pantalla “Adios (nombre de usuario)”
- Error, error: Mostrara por pantalla que la tarjeta no está asignada a ningún usuario.

Tras mostrar el mensaje, salimos del estado actual de la máquina de estados, para entrar de nuevo en el bucle de NodeMcu.

```
      //respuesta del servidor
      if (estado.equals("200")) { //200
        lcd.print("Bienvenido ");
        lcd.setCursor(1, 1);
        lcd.print(nombreUsuario);

      } else if (estado.equals("201")) {//201
        lcd.print("Adios ");
        lcd.setCursor(1, 1);
        lcd.print(nombreUsuario);
        //USE_SERIAL.print(nombreUsuario);
      }  else {
        lcd.print("NFC no");
        lcd.setCursor(1, 1);
        lcd.print("asignado");
      }

    } else {
      // if you didn't get a connection to the server:
      Serial.println("connection failed");
    }

  }

  delay(3000);
  reiniciar();
}
```

Ilustración 79. Código respuestas.

4.4.4.2. Página web

En lo que respecta a la parte de comunicación, primero explicare cómo se van a comunicar el módulo y la página web o servidor. La comunicación será tanto por parte del módulo hacia el servidor como viceversa, y estas utilizaran distintas técnicas, las cuales pasare a explicar a continuación.

Se realizara a través del Protocolo de transferencia de hipertexto o HTTP, que es un protocolo de comunicación que permite las transferencias de información en la World Wide Web. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. El cliente realiza una petición enviando un mensaje, con cierto formato al servidor. El servidor le envía un mensaje de respuesta (Ilustración 80).

Ilustración 80. Protocolo HTTP.

En el contenido de los mensajes vendrá incluido un JSon es un formato de texto sencillo para el intercambio de datos.

JSON está construido sobre dos estructuras:

- Una colección de pares de nombre / valor. Se realiza como un objeto, registro, estructura, diccionario, tabla hash, lista con clave o matriz asociativa.
- Una lista ordenada de valores. Se realiza como una matriz, vector, lista o secuencia.

En la Ilustración 81 se muestra un esquema de su uso.

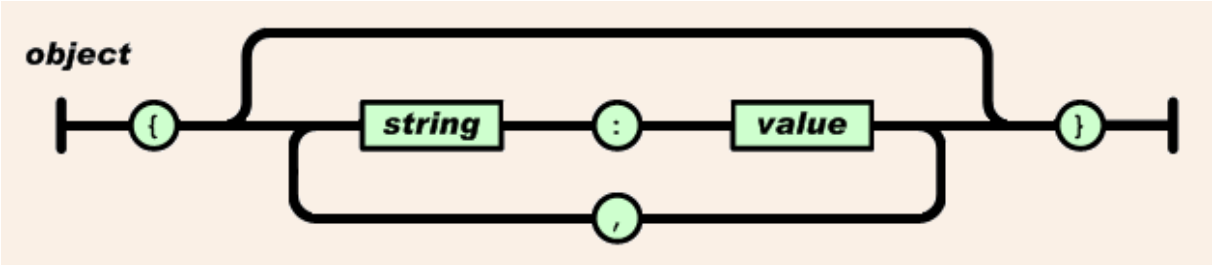


Ilustración 81. Estructura JSon.

Una vez explicado el formato que seguirá la comunicación, empezare por la parte de comunicación de la página web, donde he creado un nuevo controlador

llamado Api_controller el cual será el encargado de comunicarse con el módulo de NodeMcu, por lo que en cada método se debe enviar una respuesta a dicho módulo.

Existen tres métodos, dos encargados de dar de alta al usuario (Codigo_get y Asignar_get) y uno para guardar las entradas/salidas de los usuarios (Tarjeta_get).

Método Código_get()

En la Ilustración 82 se muestra el código de este método el cual se encarga de recibir un DNI leído a través del teclado por el sistema NodeMcu, lo valida con la base de datos a través del modelo usuario_model para comprobar si este usuario esta dado de alta en la página web devolviendo las siguientes respuestas:

- Si el usuario se registra por primera vez devuelve “nuevo”.
- Si el usuario ya tiene asignada una tarjeta NFC devuelve “registrado”.
- Si el usuario no se encuentra en la base de datos devuelve “error”.

```php
/**
* Devuelve una respuesta json con información sobre el usuario
* dependiendo de si es la primera vez que utiliza el sistema
*  y de si tiene asignada  una tarjeta NFC.
* Nota: el sistema arudino hace una llamada a la pagina con el
* dni insertado en el sistema.
*
* @access public
* @param String codigo
* @return json con id de usuario y codigo de respuesta
*/

public function codigo_get()
{
  if(!$this->get('codigo'))
  {
      $array = array('usuario'=> "vacio", 'codigo'=>"400");
      $this->response($array);
  }

  $codigo=$this->get('codigo');
  $usuario=$this->usuario_model->getUsuarioDni($codigo);

      if($usuario){ //comprobamos si hay algun usuario con ese codigo
          if($this->usuario_model->getTarjeta($usuario->id)){ // comprueba si tiene tarjeta asignada


              $array = array('usuario'=> $usuario->id, 'codigo'=>"registrado");
              $this->response($array);

          }else{
              $array = array('usuario'=> $usuario->id, 'codigo'=>"nuevo");
              $this->response($array);
          }

      }else{
          $array = array('usuario'=> "error", 'codigo'=>"error");
          $this->response($array);
      }
  }
```

Ilustración 82. Función codigo_get().

Método Asignar_get()

Este segundo método que he desarrollado es el que se muestra en la Ilustración 83, el cual aparece después de que el usuario interactúe con el sistema NodeMcu e indique a través del teclado que desea asignar un dispositivo NFC, de manera que se encarga de asignar la tarjeta o llavero NFC al usuario que anteriormente ha introducido su DNI en el sistema. Para realizar esta operación comprueba que el código NFC no sea utilizado ya por otro usuario de forma que:

1. Si el código está libre, se lo asigna al usuario y devuelve un Json con “true”.

2. Si el código ya está asignado a un usuario no realiza ninguna inserción y devuelve “false”.

```php
/**
 * Comprueba si puede asignar un nfc a un determinado alumno
 *
 *
 * @access public
 * @param String usuario , String tarjeta
 * @return json con el usuario y el codigo de respuesta
 */

public function asignar_get(){

    if(!$this->get('tarjeta') && !$this->get('usuario')) {
        $array = array('usuario'=>"", 'codigo'=>"400");
        $this->response($array);

    }

    $id=$this->get('usuario');
    $tarjeta=$this->get('tarjeta');
    $comprueba=$this->usuario_model->comprobarTarjeta($tarjeta);
    if(is_null($comprueba)){
        $this->usuario_model->insertarTarjeta($tarjeta,$id);
        $array = array('usuario'=> "",'codigo'=>"true");
        $this->response($array);


    }else{

        $array = array('usuario'=> "", 'codigo'=> "false");
        $this->response($array);
    }
}
```

Ilustración 83. Función asignar_get()

Método Tarjeta_get()

Este método recibe del módulo de identificación el código asociado a su dispositivo NFC y comprueba en la base de datos que exista un usuario con el código asociado a esa tarjeta. De ser así, obtiene la fecha, día y hora del mes y los introduce en la tabla horario y se los asocia a dicho usuario. Para marcar el fichaje como una entrada o una salida, obtengo de la base de datos el número de instancias de dicho día por ese usuario y compruebo si ese número es para o impar. Si es par significa que es una Entrada y de lo contrario una Salida como se muestra en la Ilustración 84.

```php
/**
 * Comprueba si puede asignar un nfc a un determinado alumno
 *
 *
 * @access public
 * @param String tarjeta
 * @return json con el usuario y el codigo de respuesta
 */

public function tarjeta_get(){
    if(!$this->get('tarjeta'))
    {
        $array = array('usuario'=> "", 'codigo'=> "400");
        $this->response($array);
    }

    $tarjeta=$this->get('tarjeta');
    $usuario=$this->usuario_model->getUsuarioTarjeta($tarjeta);

                if($usuario){ //comprobamos si hay algun usuario con ese codigo
                    $fecha=getDate();
                    $num=$this->hora_model->getEntrada($usuario->id,$fecha['year'],$fecha['mon'],$fecha['mday']);
                    if($num % 2==0){
                        $this->hora_model->insertar($usuario->id,FALSE);
                        $array = array('usuario'=> $usuario->nombre, 'codigo' => "entrada");
                        $this->response($array);
                        //$this->response($usuario->nombre, 200);
                    }else{
                        $this->hora_model->insertar($usuario->id,TRUE);
                        //$this->response($usuario->nombre, 201);
                        $array = array('usuario'=> $usuario->nombre, 'codigo'=> "salida");
                        $this->response($array);
                    }

                }else{
                    //$this->response("NFC no asignada", 401);
                    $array = array('usuario'=> "", 'codigo'=> "error");
                    $this->response($array);
                }

            }

        }
```

Ilustración 84. Método Tarjeta_get

4.4.5. Pruebas

Podría decirse que está es la última fase de prueba del sistema, ya que toda la parte de implementación esta creada, es por eso que el resultado lo mostrare en el siguiente incremento ya que las pruebas serán las mismas, la única variante será que el módulo estará dentro de la carcasa que se haya fabricado.

4.5. Quinto Incremento

4.5.1. Análisis

Para el análisis de este incremento pasare a enumerar al igual que en el caso de la página web tanto los requisitos funcionales como los no funcionales que el sistema debe cumplir.

4.5.2.1. Requisitos Funcionales

Los requisitos funcionales, que como se ha explicado anteriormente, son aquellos que describen el comportamiento del sistema.

Requisitos funcionales para el sistema de identificación:

1. Cualquier usuario dado de alta en la página web podrá asociar una tarjeta NFC a través del teclado del sistema.
2. Cualquier usuario con una tarjeta NFC asociada podrá pasar asistencia a través del sistema de fichaje.
3. El sistema debe ser capaz de mostrar al usuario por la pantalla el resultado de las diferentes operaciones, tanto la de pasar asistencia, como la de asignar un dispositivo NFC.
4. El usuario debe poder borrar en cualquier momento la cifra introducida por el teclado en caso de equivocación.

4.5.2.2. Requisitos No Funcionales

Son aquellos como ya se ha explicado que son ajenos al usuario, relacionados con las características del funcionamiento.

Requisitos No funcionales para el sistema de identificación:

1. El tiempo de respuesta del sistema debe ser aceptable.
2. Debe estar disponible las 24 horas del día.
3. No pueden existir dos usuarios con la misma tarjeta asociada.

4.5.2. Diseño

Para empezar con el diseño de la carcasa, este debe ser lo suficientemente amplio para que quepan todos los componentes y sus conexiones. Teniendo en cuenta que el teclado y la pantalla deben estar por fuera para que el usuario

interactúe con ellos, creare una base donde estarán los diferentes dispositivos y el propio NodeMcu con sus conexiones(cables), de forma que no puedan ser vistas por los usuarios, y otra exterior que hará a su vez de tapa donde estarán el teclado y la pantalla led.

A la base le añadiré diferentes habitáculos para que queden sujetos todos los módulos. La carcasa que hará de base tendrá dos orificios, uno para la conexión ethernet y otro para la alimentación de NodeMcu. Las carcasas se unirán por tornillos y tuercas de 6 milímetros. Tras medir los diferentes módulos y utilizando el programa 3D builder he obtenido el resultado que se muestra en la Ilustración 85 y la Ilustración 86.

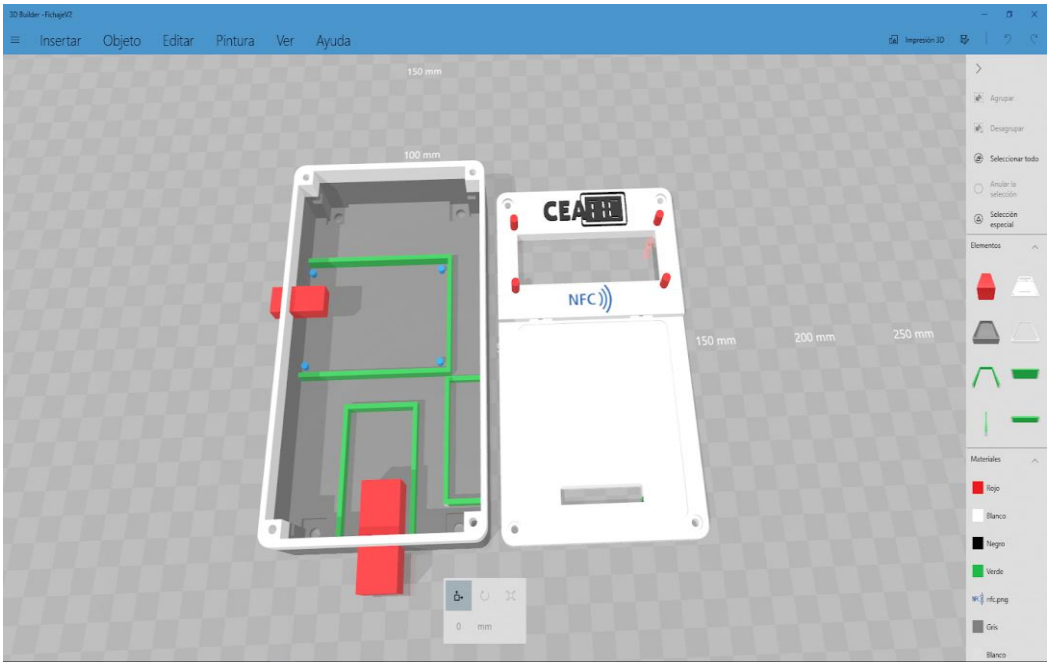


Ilustración 85. Modelo carcasa vista superior.

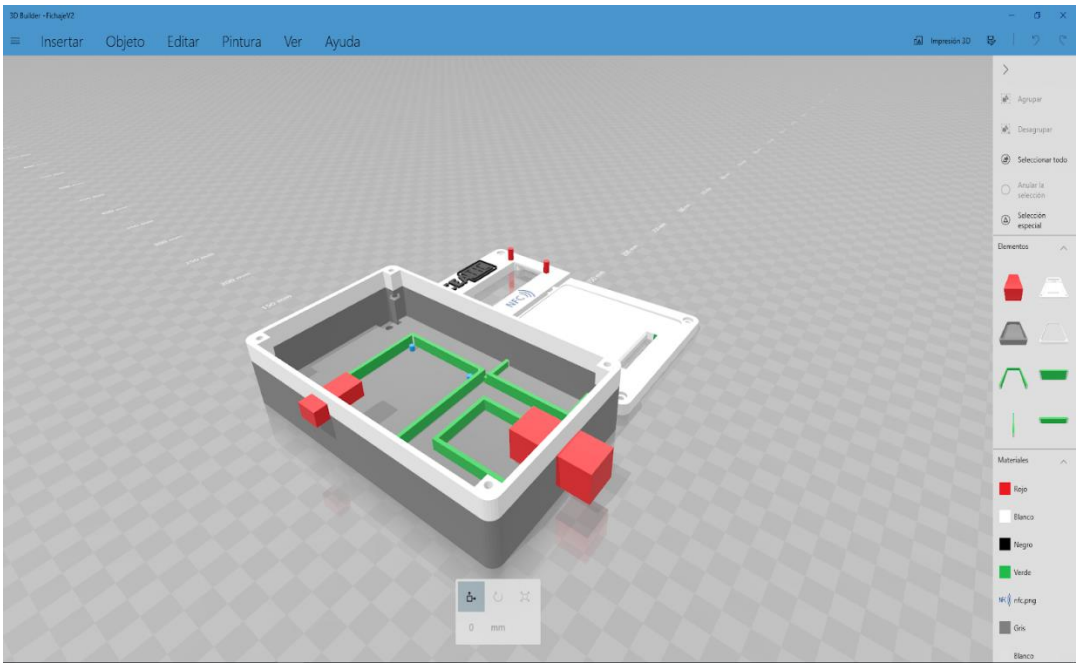


Ilustración 86. Modelo carcasa vista perfil.

4.5.3. Implementación

La implementación de este incremento consistirá en crear el modelo con una impresora 3D y en soldar las conexiones de los dispositivos para evitar fallos en un futuro.

Para la realización de la impresión he utilizado la impresora Prusa i3 Hephestos de BQ de la Ilustración 87, que es una impresora 3D de fabricación por deposición de filamento fundido de código abierto. El bajo costo y la facilidad de construcción y modificación de la impresora la han convertido en una popular herramienta.

Ilustración 87. Impresora Prusa i3 Hephestos.

He realizado varias impresiones de la carcasa ya que muchas veces había fallos durante la impresión y el modelo salía dañado y teniendo en cuenta que la duración de la impresión de cada parte de la carcasa era de alrededor de unas 8 horas, fueron varios días los que tarde en obtener un modelo aceptable.

Mientras el modelo se imprimía, pase a la parte de soldar las conexiones NodeMcu utilizando el soldador disponible en el CEATIC. He utilizado una placa PCB 8x4 que servirá de base para NodeMcu. El principal objetivo de esto es el de ahorrarnos conexiones dado que gracias al estaño las pistas enlazadas con el comparten la señal.

Una vez terminado con la impresión y la parte de soldadura, he procedido a introducir NodeMcu en la carcasa utilizando por un lado tornillos m3 para el led y cinta aislante de doble cara para adherir los diferentes componentes a la carcasa. El resultado puede verse en la Ilustración 88, la Ilustración 89, la Ilustración 90 y la Ilustración 91.

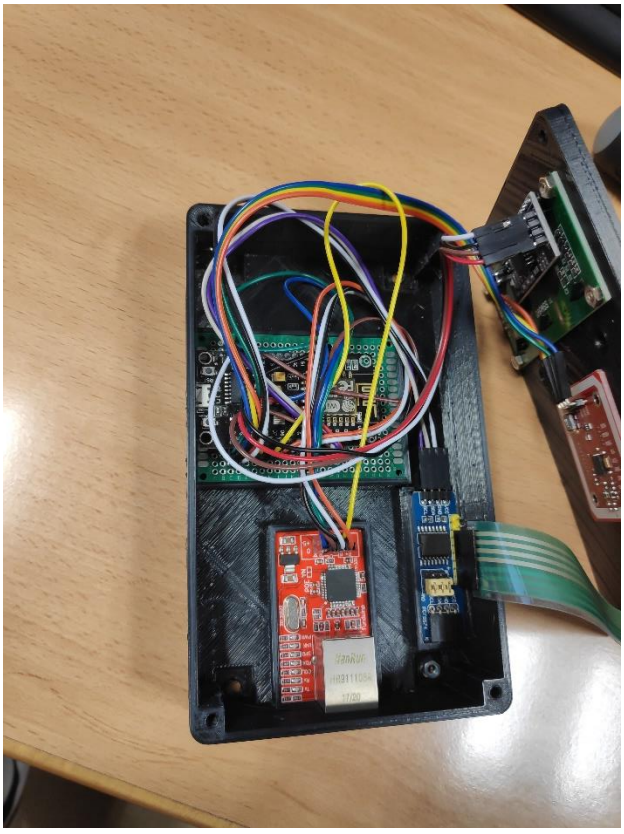


Ilustración 88. Resultado Módulo vista 1.

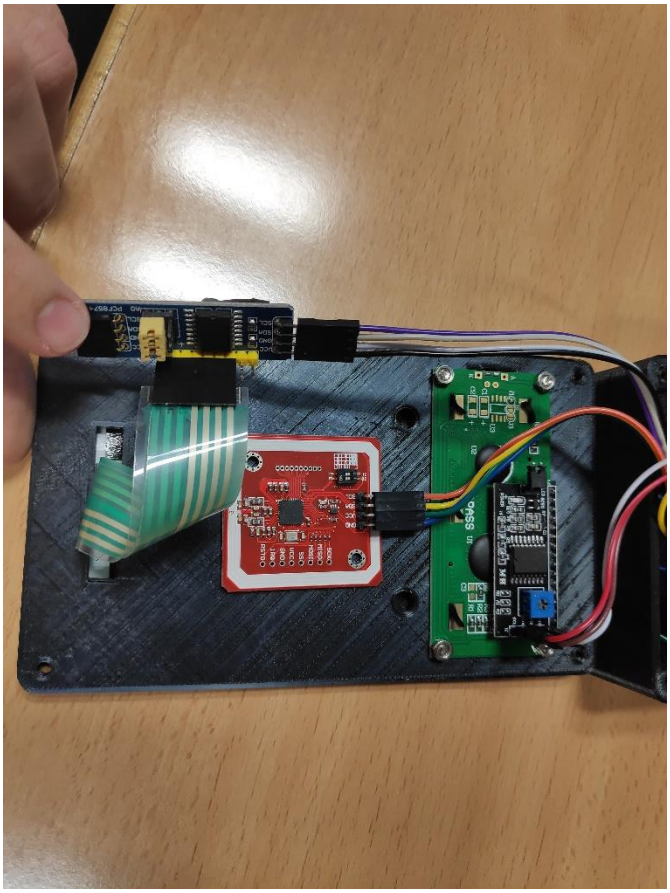


Ilustración 89. Resultado del módulo vista 2.

Ilustración 90. Resultado módulo vista 3.

Ilustración 91. Resultado del módulo vista 4.

4.5.4. Pruebas

En este incremento se le pedirá al usuario que pase asistencia para comprobar su funcionamiento.

Para ello primero deberá introducir su DNI en el sistema a través del teclado, interactuar con el sistema y asignar por primera vez su tarjeta NFC y que durante unos días el usuario pase asistencia durante su jornada laboral.

En la Ilustración 92 se muestra el resultado de introducir el DNI del usuario.

Ilustración 92. Asignación NFC parte 1.

En la Ilustración 93 se muestra el mensaje las opciones para asignar el dispositivo NFC.

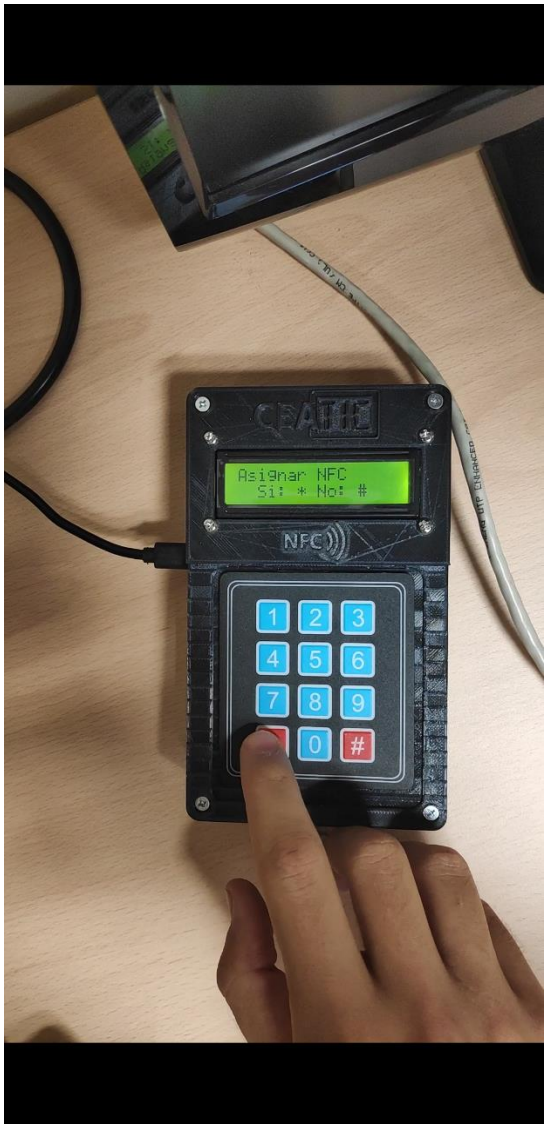


Ilustración 93.Asinacón NFC parte 2.

En la Ilustración 94 se muestran el resultado de pasar el dispositivo NFC, primero como si fuera una entrada y después una salida.

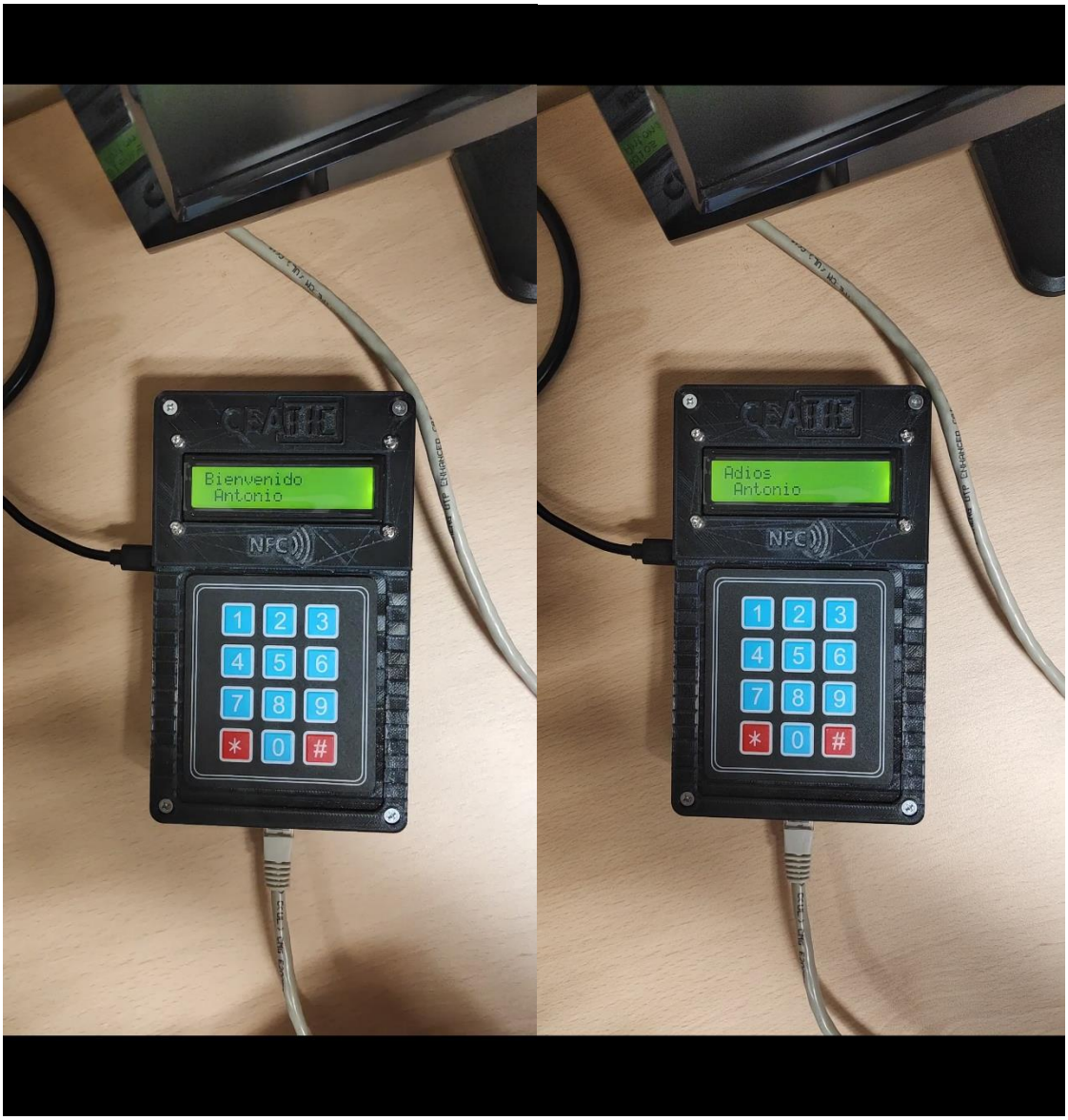


Ilustración 94. Respuesta del módulo.

En la Ilustración 95 se muestra en resultado de varios días pasando la asistencia.

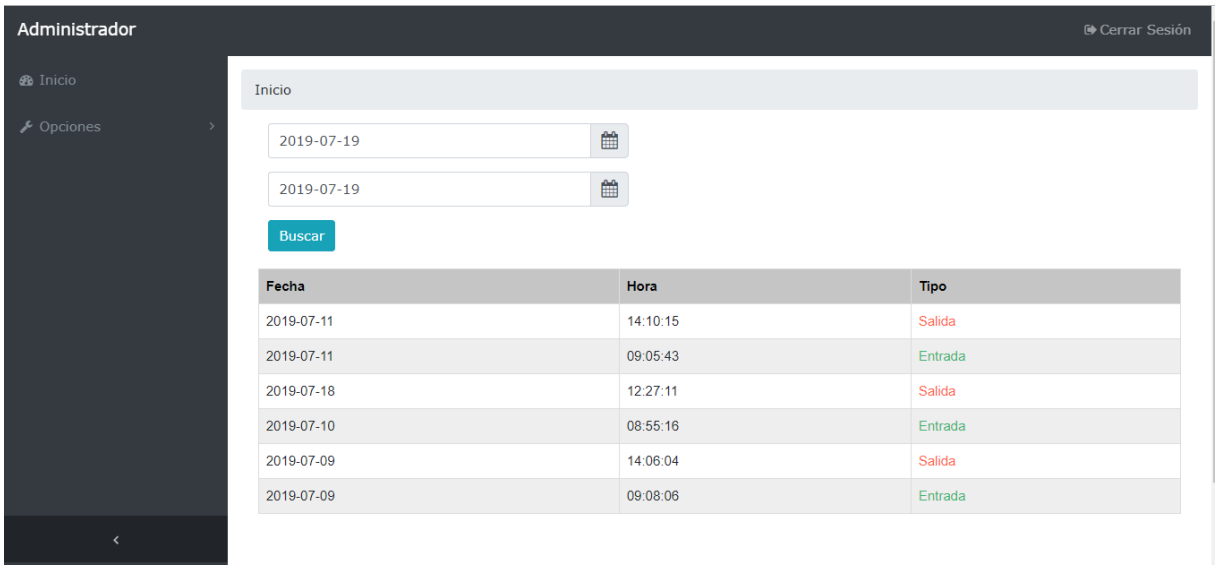


Ilustración 95. Resultado horario.

5. Capítulo 5. Conclusiones y Mejoras

Una vez realizada la fase de pruebas del sistema, he sacado en claro las siguientes conclusiones:

- Los tiempos dispuestos por el diagrama de Gantt han sido suficientes para realizar los objetivos de nuestro proyecto, aunque en algunos incrementos se han realizado en un número menor de horas, pero no de días.
- La adaptación responsive de nuestra página se ha realizado de manera automática gracias al tema que se ha escogido.
- Ha supuesto un gran reto toda la parte asociada a la creación del microprocesador, ya que es algo diferente a lo que se ha estudiado a lo largo de la carrera.
- En la parte de conexión de los diferentes componentes del nodeMcu ha sido una de la más arduas del proyecto, ya que físicamente cada uno debe ir conectado a unos pines específicos y en ocasiones existía interferencias entre ellos además de que algunos funcionan a un voltaje diferente.
- En lo que respecta a la creación de la carcasa 3d, me ha sorprendido la facilidad a la hora de crear un modelo y pasarlo a la impresora 3d. El único problema era el tiempo de impresión ya que al ser una impresora que no es de gama alta, los tiempos de impresión eran muy altos, ya al ser la calidad baja, en muchas ocasiones había que volver a repetir el proceso de impresión por fallos en el mismo.

Propuesta de mejoras para realizar en el futuro:

- Que el sistema muestre el total de horas trabajadas, tanto semanales como por meses.
- Que el usuario sea capaz de poder comunicarse con su jefe de departamento o con el administrador, para comunicar posibles faltas o recuperaciones de horas.

- Que tanto el jefe de departamento como al administrador dispongan de alertas para que cuando un usuario no ha asistido al trabajo o al cabo de la semana no llega al mínimo de horas sepan de esa información.
- Que el sistema de identificación sea capaz de mostrar la hora por pantalla a la que se realiza el fichaje.
- Mostrar información en el perfil de cada usuario sobre si tiene asociado un dispositivo NFC

Bibliografía

- Llamas, L. (9 de Julio de 2019). *Luis Llamas*. Obtenido de https://www.luisllamas.es/NodeMcu-NFC-pn532/
- Nielsen, J. (2000). *Usabilidad. Diseño de sitos Web.* Prentice Hall.
- Pfleeger, S. L. (s.f.). (2002) *Ingeniería de software, teoría y practica.* Prentice Hall.
- SILBERSCHATZ, A. (2007). *Fundamentos de bases de datos.* S.A. MCGRAW-HILL / INTERAMERICANA DE ESPAÑA.
- *php*. (17 de Julio de 2019). Obtenido de https://www.php.net/
- *NodeMcu*. (17 de Julio de 2019). Obtenido de https://www.NodeMcu.cc/
- CodeIgniter. (17 de Julio de 2019). *CodeIgniter.com*. Obtenido de https://www.codeigniter.com/user_guide/
- *stackoverflow*. (17 de Julio de 2019). Obtenido de https://es.stackoverflow.com/
- Valle, L. d. (17 de Julio de 2019). *programarfacil.com*. Obtenido de https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/
- *w3schools*. (17 de Julio de 2019). Obtenido de https://www.w3schools.com/html/
- *Wikipedia*. (17 de Julio de 2019). Obtenido de https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

Anexos

Anexo I. Manual de instalación del sistema

En este manual se detallara paso a paso, el proceso de instalación que se deberá llevar a cabo para poder utilizar este software. Para empezar se deberá instalar en nuestro sistema el software Xampp que se puede descargar de forma gratuita a través de este enlace: https://www.apachefriends.org/es/index.html

Una vez dentro, hay que seleccionar la versión apropiada para nuestro sistema operativo.

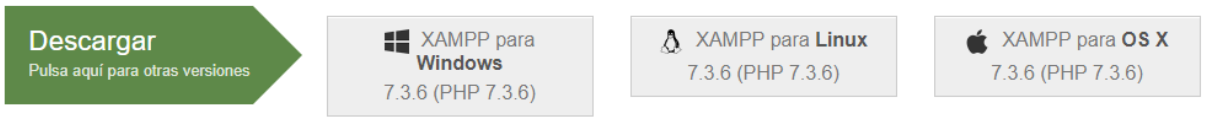


Ilustración 96. Versiones Xampp.

Tras realizar la descarga, pasamos a su instalación. Primero hay que elegir donde queremos que se realice la instalación en nuestro sistema, recomiendo guardarlo en la dirección que se asigna por defecto, y pulsar siguiente. Las siguientes ventanas son para escoger distintas funcionalidades que ofrece este software, dejaremos las que vienen por defecto y pulsaremos siguiente en todas ellas hasta llegar a la ventana final donde pulsaremos finalizar.

Tras realizar la instalación en nuestro sistema, debemos introducir los ficheros para la base de datos y la página web.

Base de Datos

Para introducir la base de datos primero hay que iniciar el programa xampp, ya que la única manera de acceder a ella es de forma local. Una vez iniciado el programa hay que activar las opciones de Apache y MySQL a través del panel de control, como se ve en la Ilustración 97.

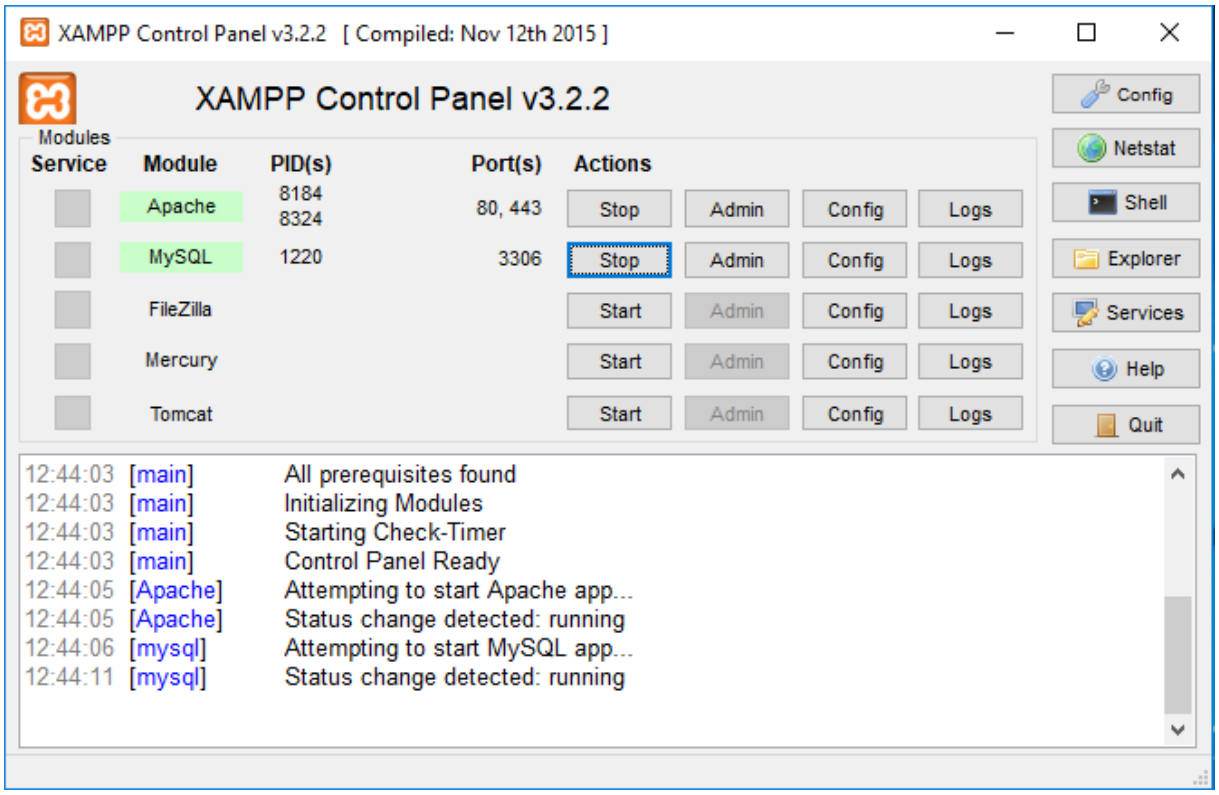


Ilustración 97. Panel Control Xampp.

He de añadir que suele existir algún tipo de problema con el módulo de Apache ya que algunos programas como Skype utilizan el mismo puerto para su funcionamiento, la solución es cerrar el programa que nos ocasiona la interferencia o re direccionar puertos. Pero como consejo personal optaría por la primera opción.

Una vez en funcionamiento hay que acceder al panel de control de la base de datos mediante la url: localhost/phpmyadmin que introduciremos en nuestro buscador. Una vez allí hay que crear una nueva base de datos a la que llamaremos control.

Tras crearla hay que importar el archivo control.sql pulsando la opción importar del panel y seleccionando el archivo que tenemos guardado en nuestro sistema. Las diferentes opciones que aparecen se dejaran por defecto y pulsaremos el botón continuar.

Como se ve en la imagen ya se pueden ver las diferentes tablas con los datos que se han introducido para utilizar el sistema.

Página Web

Para la Página web es necesario incluir el proyecto dentro de la carpeta Htdocs de Xampp.

El primer paso es ir al directorio donde hemos instalado Xampp, si no lo hemos modificado en la instalación, por defecto es C:Xampp. Una vez allí nos metemos dentro de la carpeta xampp y buscamos la carpeta htdocs. Accedemos a ella y copiamos la carpeta Asistencia.

Tras haber finalizado la instalación e incluido los ficheros de base de datos y página web. Es necesario siempre que se quiera hacer uso de la web, iniciar el programa Xampp con las opciones de Apache y MySQL activadas. A través de la url: localhost/Asistencia se puede acceder y utilizar el software.

Anexo II. Manual de usuario

Lo primero que hay que saber es que este software está hecho para ser utilizado de forma local, por lo que el primer paso para poder utilizarlo y tras la previa instalación, es activar el programa Xampp y pulsar el botón Start en las dos primeras opciones que son Apache y MySql.

Para acceder a la página web hay que hacerlo introduciendo en el navegador la siguiente dirección: localhost/Asistencia y nos direccionara a una página igual a la de la Ilustración 98.

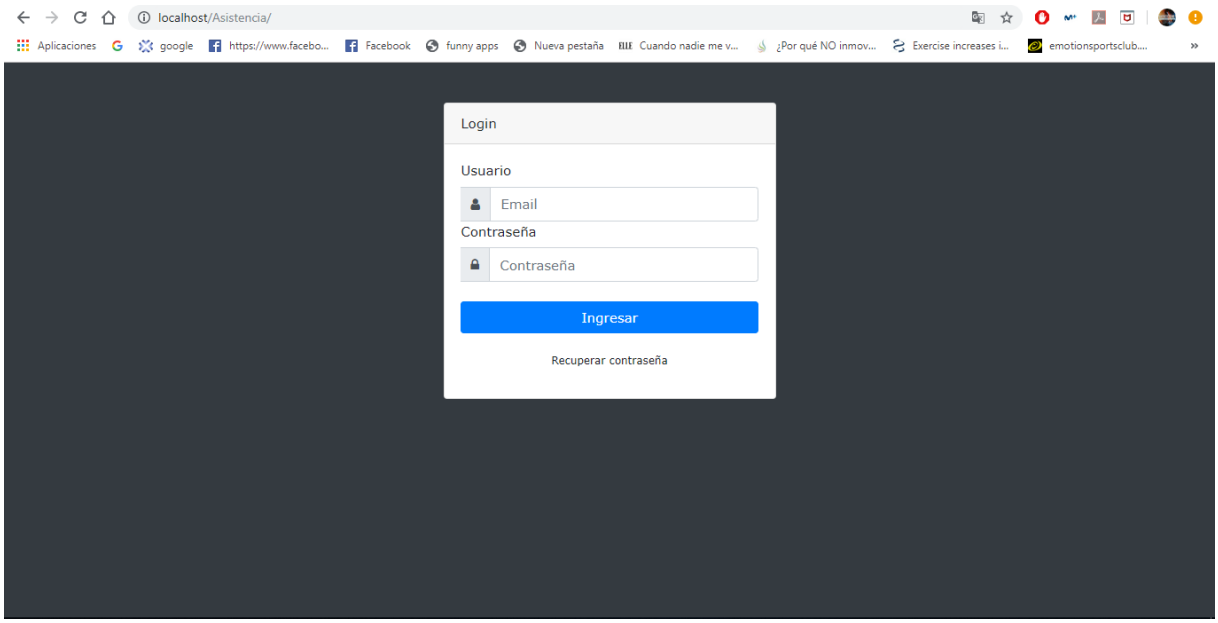


Ilustración 98. Inicio de Sesión.

Una vez dentro se puede acceder con tres correos diferentes donde cada uno tiene asociado un rol diferente. Los correos y sus respectivas contraseñas son:

- Administrador:admin@gmail.com Contraseña: admin123
- Jefe Departamento:jefeDep@gmail.com Contraseña: jefe123
- Usuario:usuario@gmail.com Contraseña: usuario123

Las opciones serán diferentes según el perfil del usuario. En la Tabla 9 se mostrara cada una de estas opciones y se indica para que tipo de usuario estarán disponibles.

	Administrador	Jefe Departamento	Usuario
Consultar perfil de un usuario.	Si	Si, si pertenece al mismo departamento.	No
Consultar perfil proprio.	Si	Si	Si
Consultar horario de un usuario.	Si	Si, si pertenece al mismo departamento.	No

Consultar horario propio.	Si	Si	Si
Crear nuevo departamento.	Si	No	No
Dar Alta/Baja Usuario.	Si	Si, si pertenece al mismo departamento	No
Crear nuevo usuario.	Si, con cualquier rol.	Si, con rol de tipo usuario.	No
Iniciar/ Cerrar Sesión.	Si	Si	Si
Cambiar Contraseña, Foto y email.	Si	Si	Si

Tabla 9. Opciones según rol.

Consultar perfil de un usuario (Administrador/ Jefe Departamento)

Para consultar el perfil de un usuario debemos estar en la página principal, y buscar dicho usuario en la tabla que nos aparece ya sea de forma manual o utilizando el buscador (Ilustración 99).

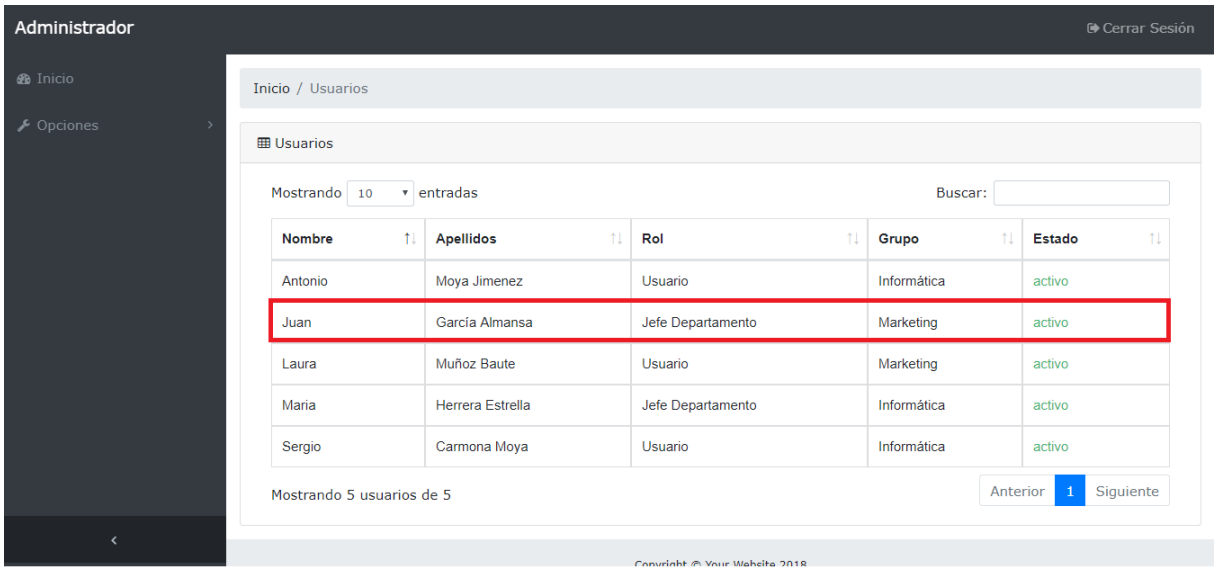


Ilustración 99. Acceder Perfil Usuario.

Una vez localizado debemos pulsar dentro de su fila y accederemos a su perfil, donde nos aparecerá una foto e información sobre dicho usuario (Ilustración 100).

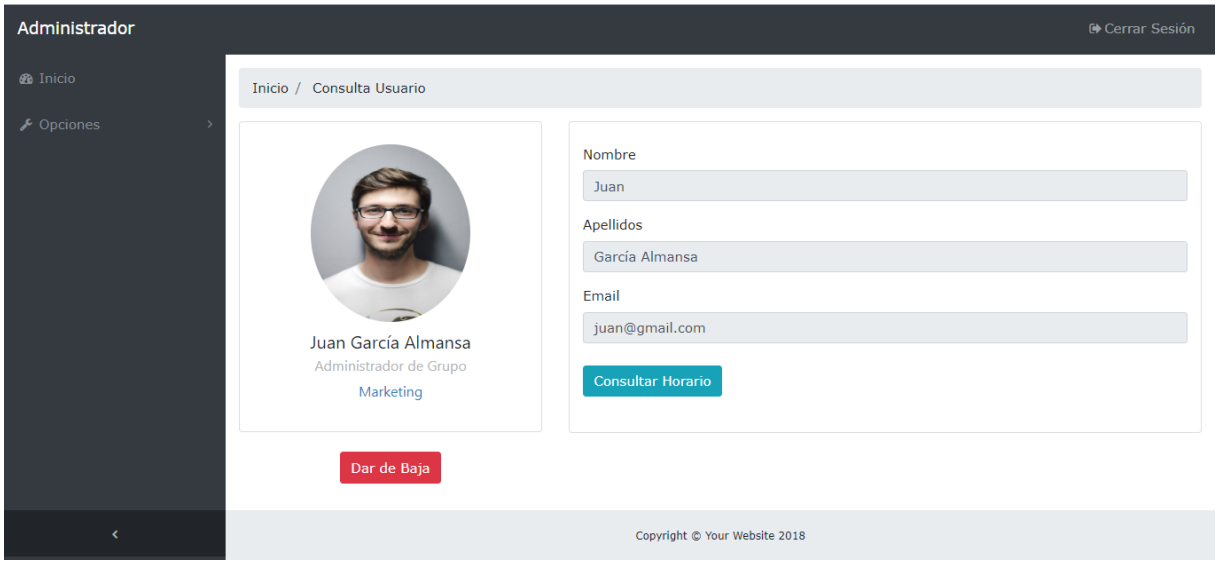


Ilustración 100. Perfil de usuario.

Crear un nuevo Departamento (Administrador)

Para acceder a esta opción hay que irse al menú lateral de la página web y pulsar la opción Gestión de Grupos (Ilustración 101).

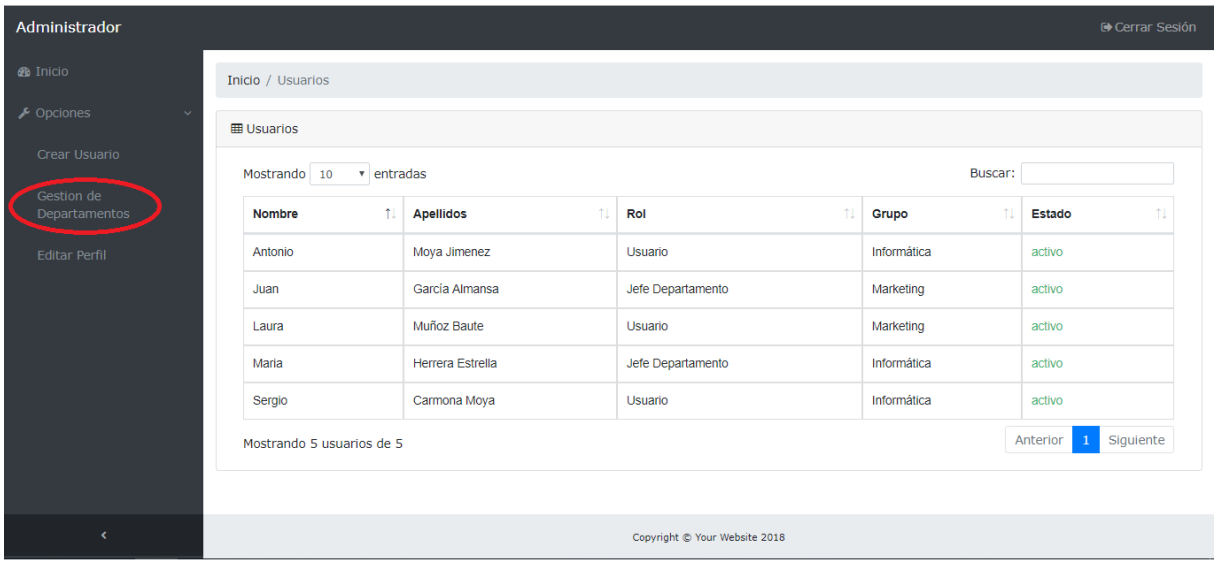


Ilustración 101. Opción Gestión de Departamentos.

Una vez dentro de esta página hay que pulsar el botón verde “Crear nuevo Departamento” y accederemos a una nueva página (Ilustración 102), donde

aparecerán un espacio para escribir el nombre del nuevo departamento y un desplegable para asociarlo a un jefe de grupo.

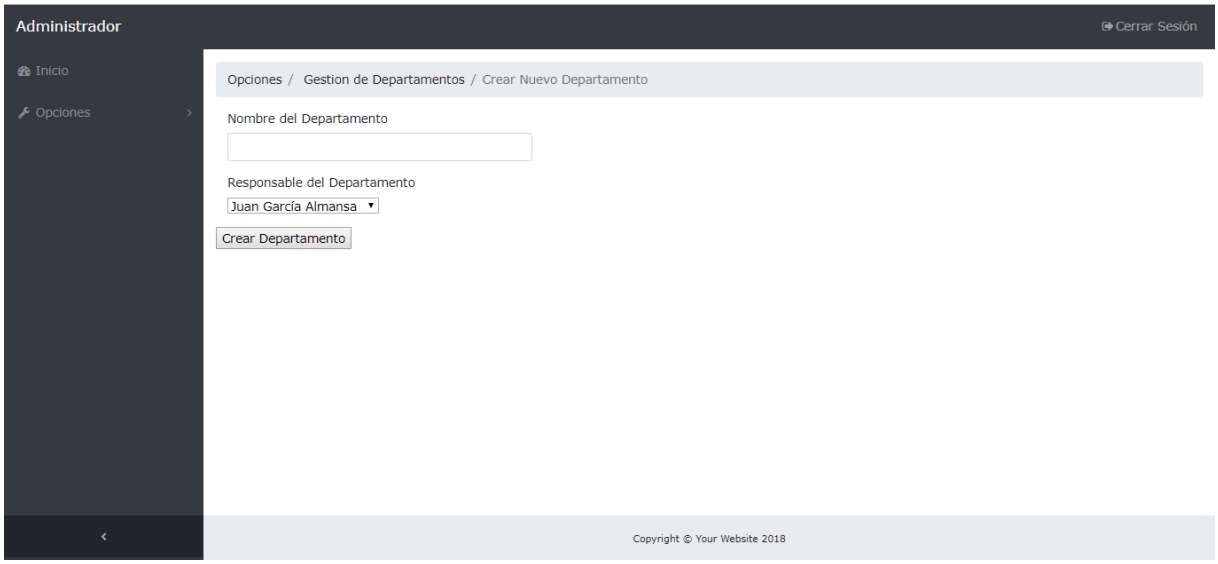


Ilustración 102. Crear nuevo departamento.

Tras rellenar las opciones pulsaremos el botón “Crear Departamento” para realizar la operación.

Crear un nuevo Usuario (Administrador/Jefe Departamento)

Para acceder a esta opción hay que irse al menú lateral de la página web y pulsar la opción “Crear Usuario” (Ilustración 103).

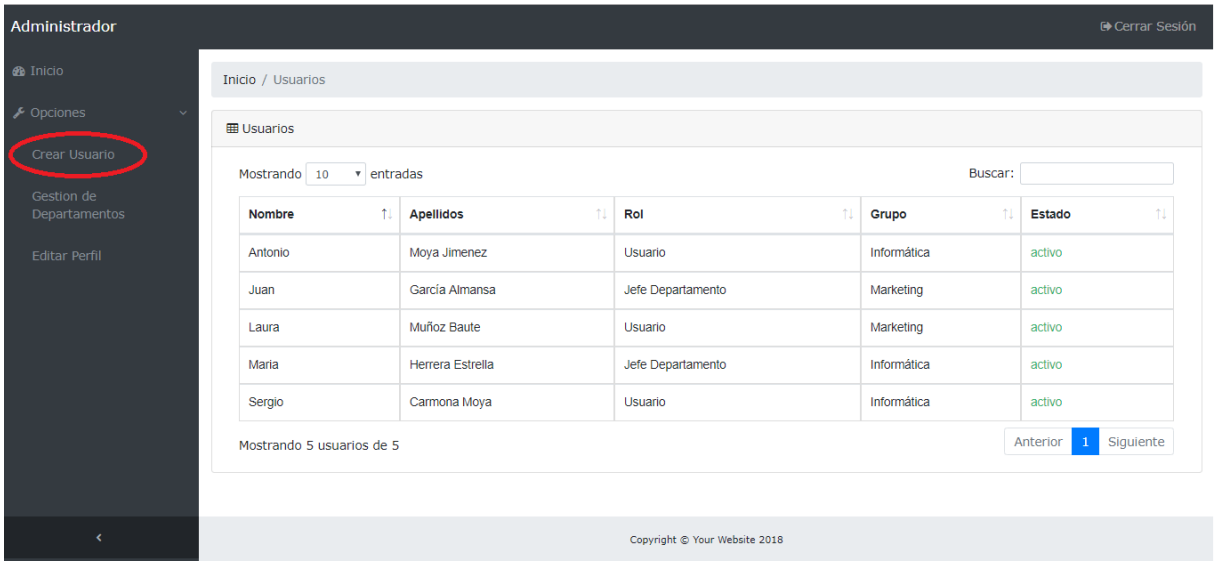


Ilustración 103. Opción crear usuario.

Una vez dentro de esta página (Ilustración 104), hay que rellenar los distintos campos y seleccionar el grupo al que pertenecerá el usuario que se va a crear y si el usuario es creado por el administrador escoger el tipo de usuario.

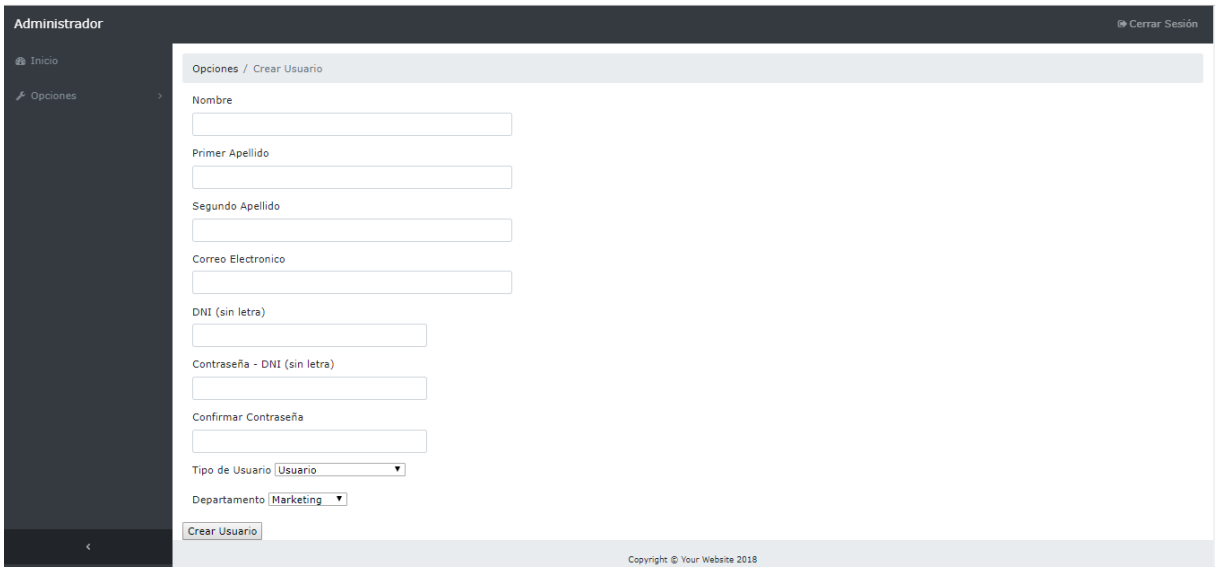


Ilustración 104. Página crear usuario.

Cuando ya se hayan rellenado las diferentes opciones pulsaremos el botón “Crear Usuario” y nos aparecerá un mensaje por pantalla indicando el resultado de la operación.

Dar de Baja/Alta Usuario (Administrador/Jefe de Departamento)

A esta opción debemos acceder a través del perfil de un usuario concreto, como se ha explicado en el apartado “Consultar un perfil de usuario”. Una vez dentro del perfil de un usuario hay que pulsar el botón rojo “Dar de Baja” o si ese usuario ha estado previamente dado de baja el botón verde “Dar de Alta” (Ilustración 105).

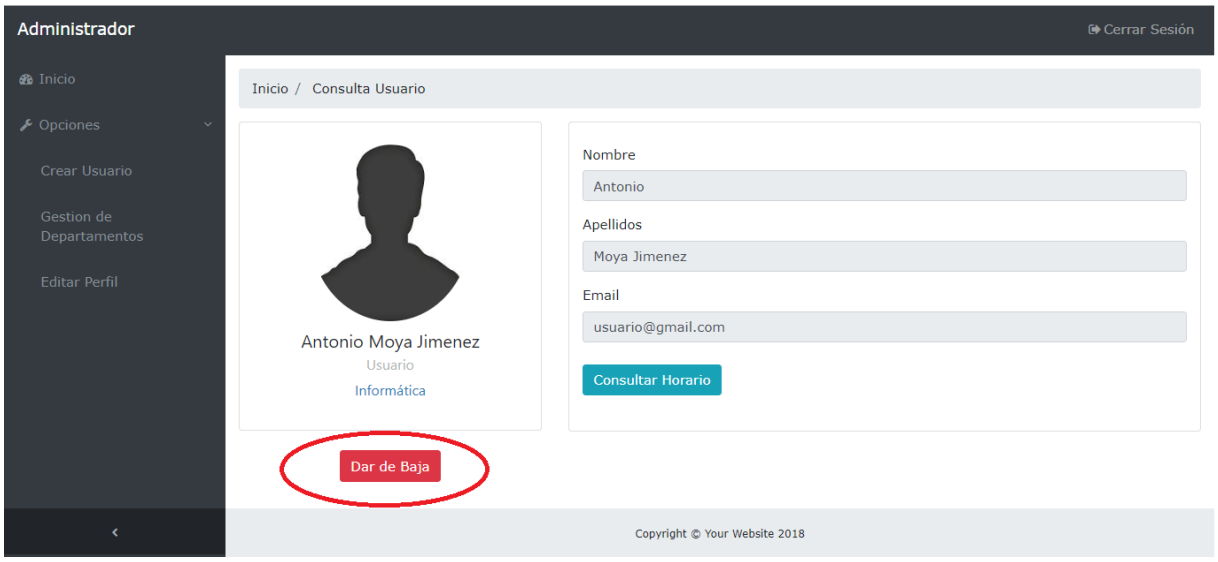


Ilustración 105. Botón dar de baja.

A continuación, nos aparecerá un mensaje (Ilustración 106) preguntado si estamos seguros de realizar dicha acción, pulsamos “Aceptar“ y el usuario pasará a cambiar de estado según la operación realizada (Ilustración 107).

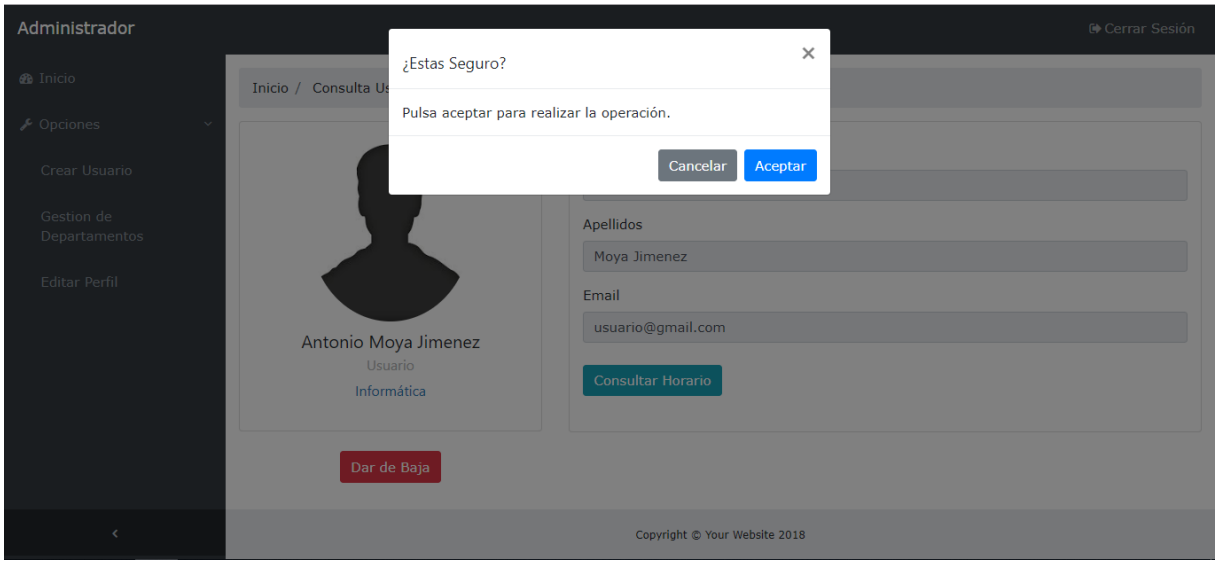


Ilustración 106. Mensaje Dar de baja/alta usuario.

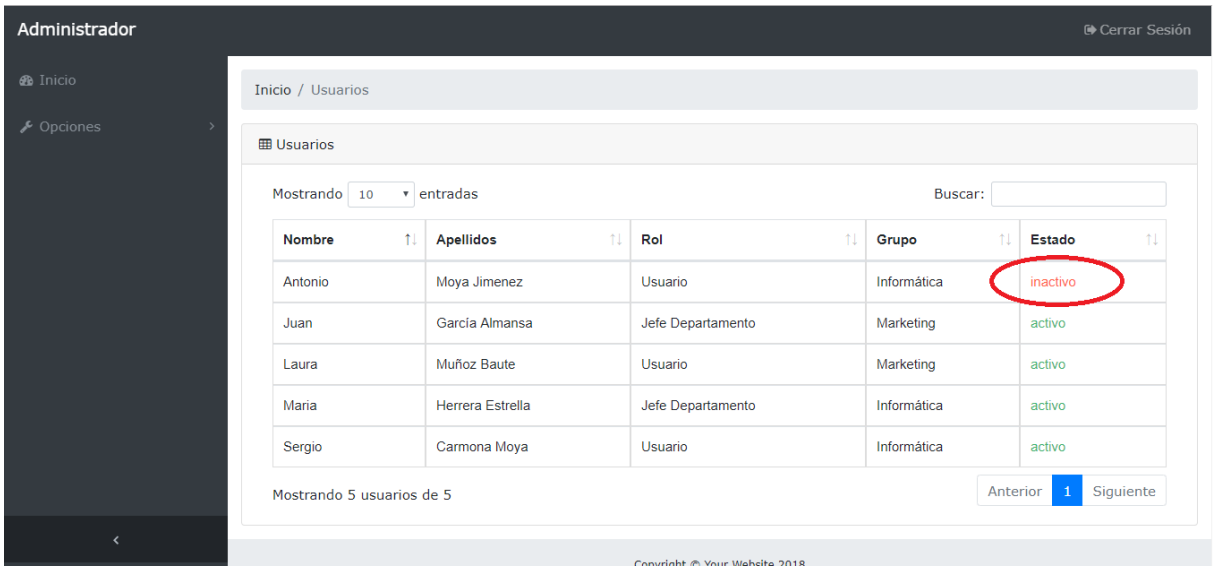


Ilustración 107. Resultado Dar de baja usuario.

Consultar Horario de un usuario (Administrador-Jefe de Departamento)

Esta opción es accesible a través del perfil de un usuario, como se ha explicado en el apartado “Consultar perfil de usuario”. Una vez dentro de esta ventana deberemos pulsar el botón “Consultar Horario” (Ilustración 108), el cual nos llevará a una nueva ventana (Ilustración 109), donde aparecerá una tabla donde se podrán ver la fechas y horas del fichaje realizado por ese usuario en ese último mes.

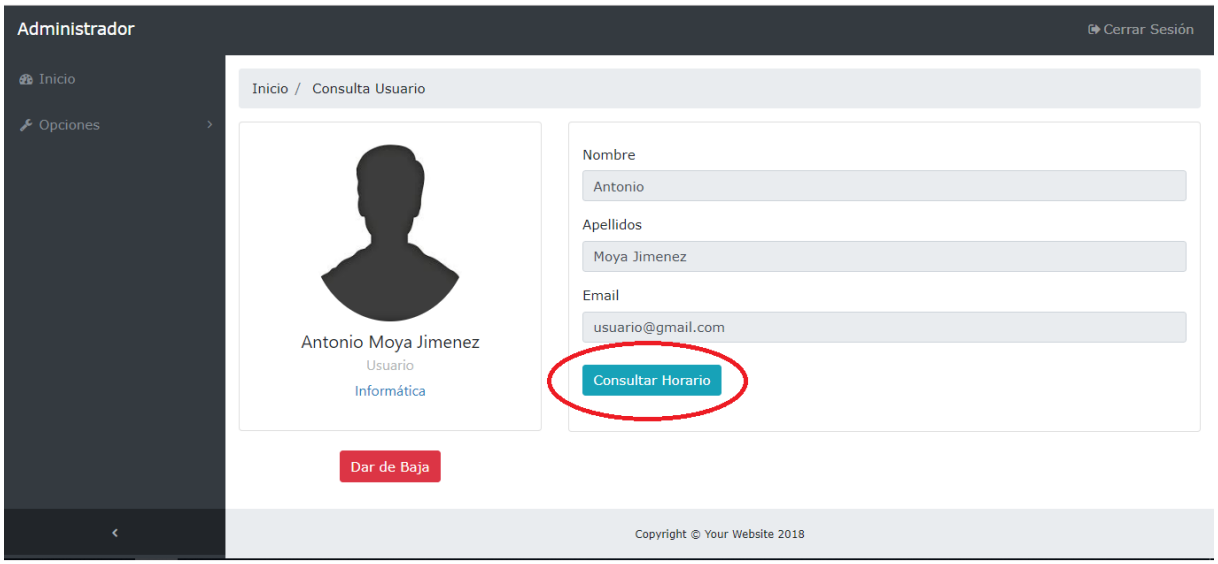


Ilustración 108. Botón Consultar Horario.

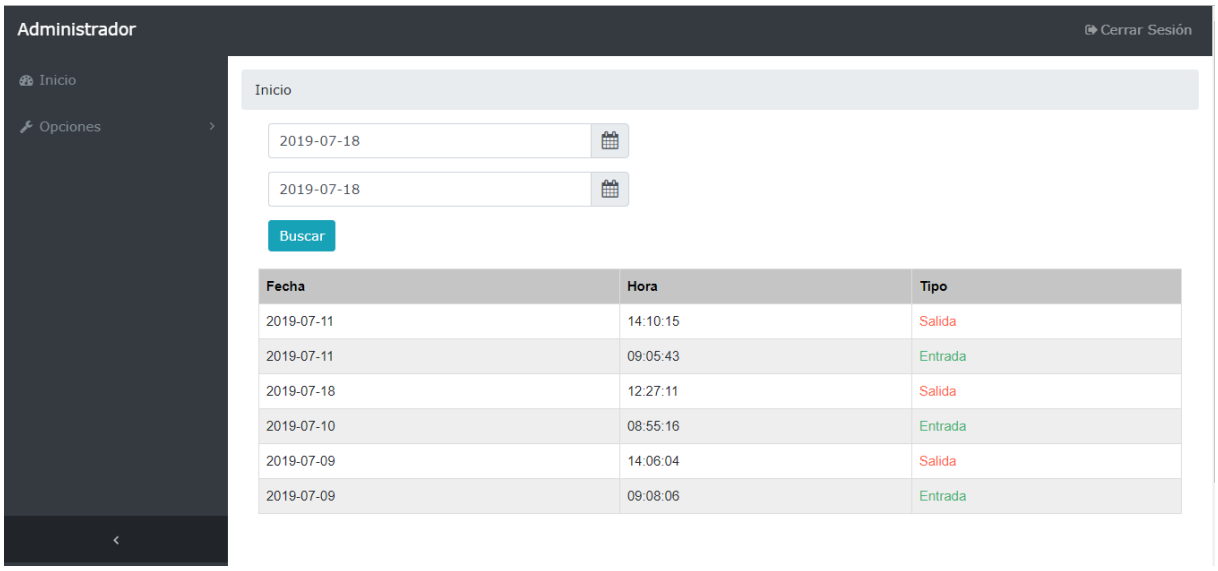


Ilustración 109. Vista conusltar Horario.

También está la opción de establecer un intervalo de fechas con el buscador que se encuentra en la parte superior de la tabla como se ve en la Ilustración 110.

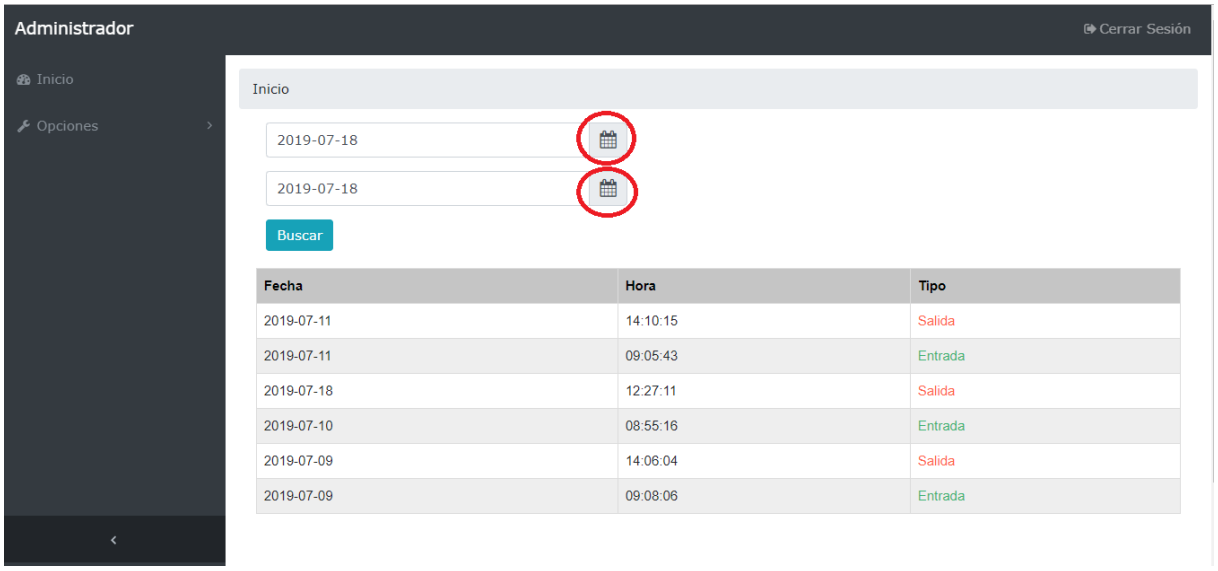


Ilustración 110. Filtrar Tabla Horario.

Cambiar contraseña

Esta opción es accesible a través de la opción “Editar Perfil” del menú lateral. Una vez dentro de la ventana de perfil de usuario debemos pulsar el botón “Cambiar Contraseña” y aparecerá un desplegable donde deberemos introducir la contraseña

actual y la contraseña por la que la queremos cambiar (Ilustración 111,Ilustración 112).

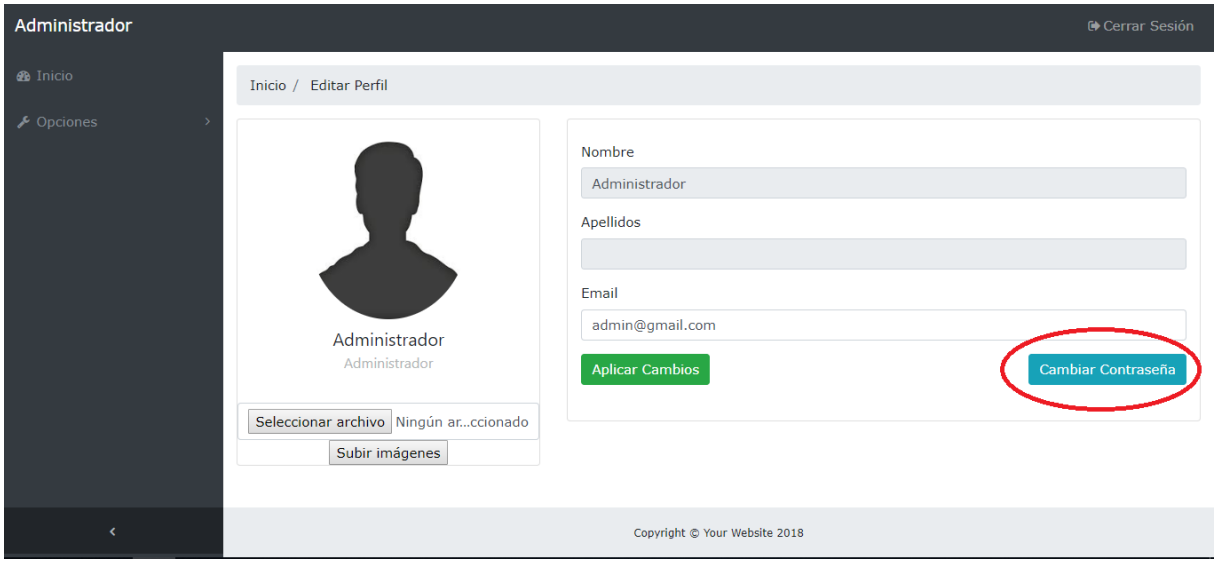


Ilustración 111. Botón Cambiar Contraseña.

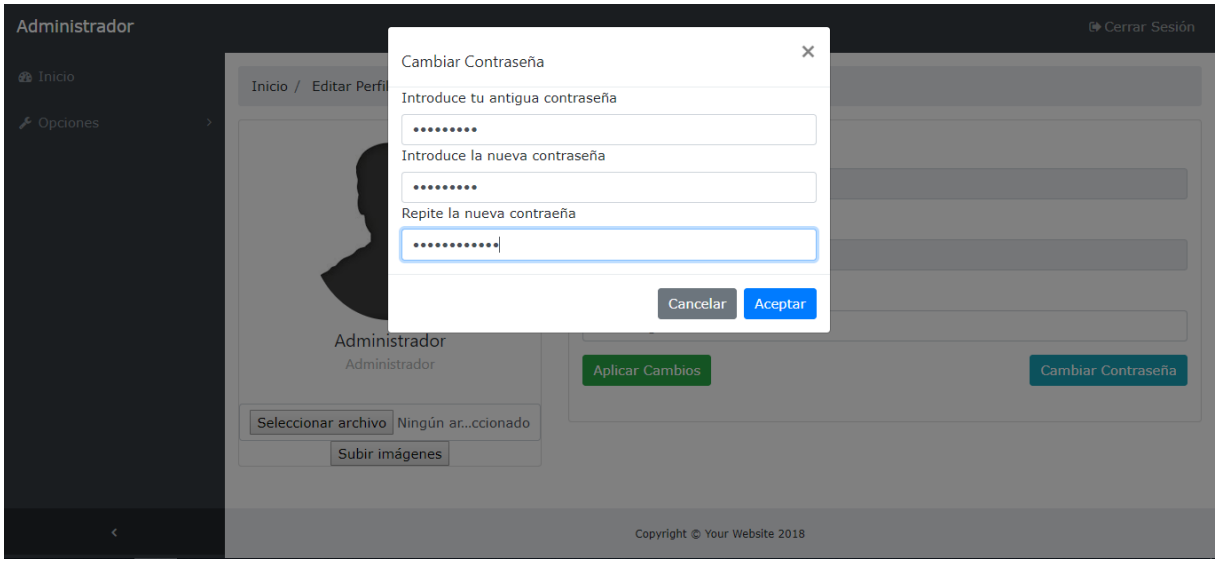


Ilustración 112. Desplegabe Cambiar Contraseña.

Tras completar los campos debemos pulsar el “Aceptar” y nos aparecerá por pantalla un mensaje indiciándonos el resultado de la operación.

Cambiar Foto de Perfil

Esta opción es accesible a través de la opción Editar Perfil del menú lateral. Una vez dentro de la ventana de perfil de usuario debemos pulsar el botón “Seleccionar Archivo” y seleccionar la nueva foto de perfil (Ilustración 113).

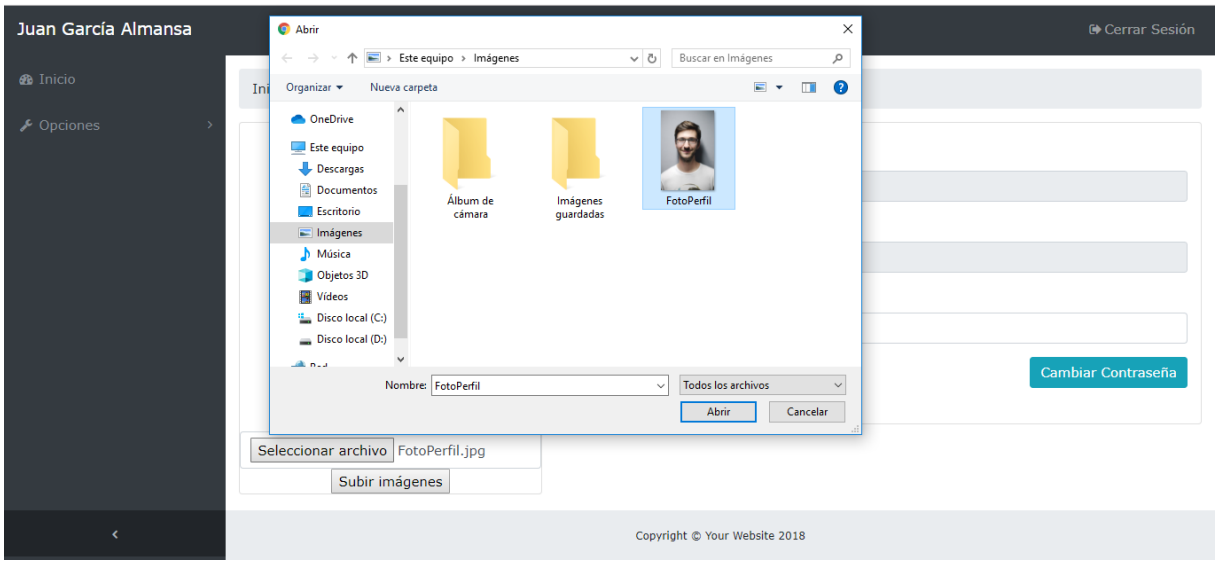


Ilustración 113. Cambiar Foto Perfil.

A continuación, hay que pulsar el botón “Subir Imagen”, por pantalla aparecerá un mensaje en caso de que la imagen no cumpla los requisitos de tamaño, en caso contrario podremos visualizar la imagen seleccionada (Ilustración 114).

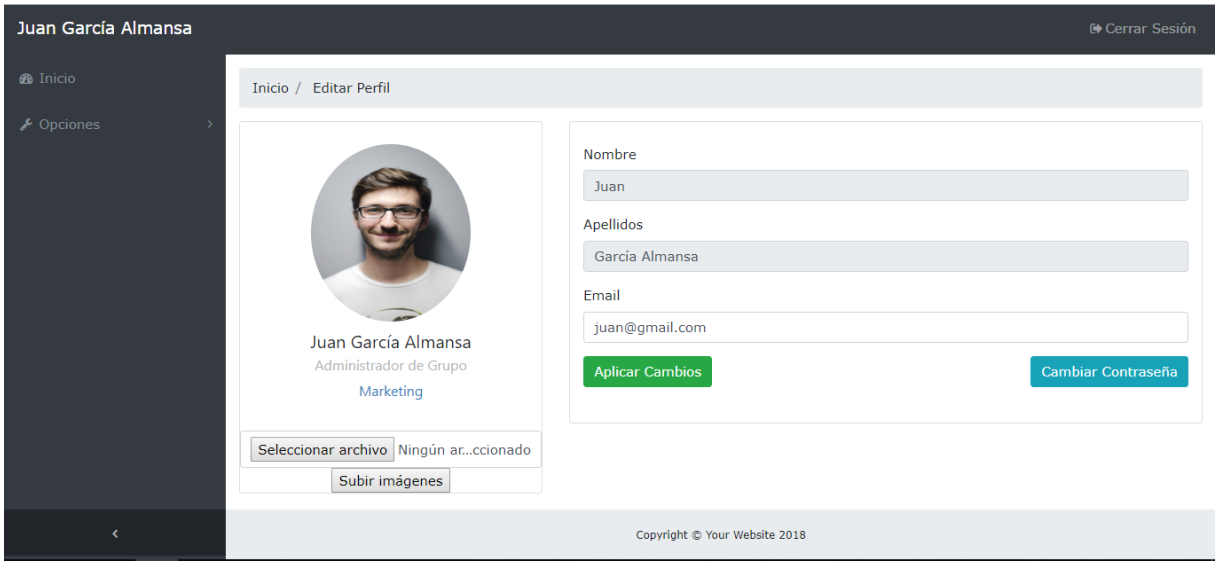


Ilustración 114. Resultado Foto Perfil.

Cambiar Email

Esta opción es accesible a través de la opción Editar Perfil del menú lateral. Una vez dentro hay que escribir el email y pulsar el botón “Aplicar Cambios” (Ilustración 115).

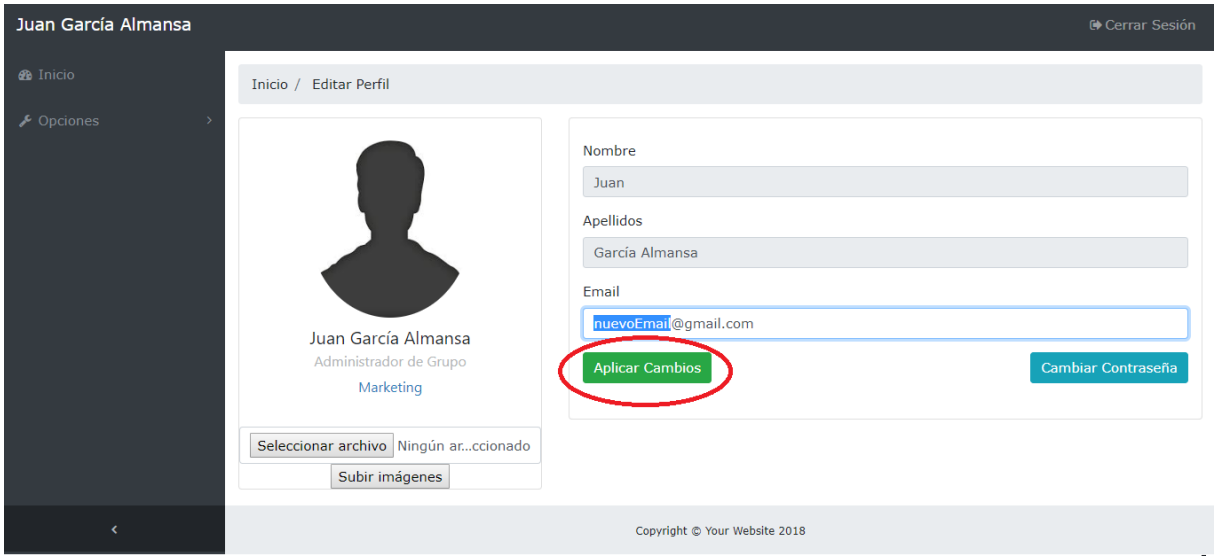


Ilustración 115. Cambiar Email.

Por pantalla aparecerá un mensaje en caso de que el email ya esté en uso por otro usuario, en caso contrario podremos visualizar el nuevo email.

Cerrar Sesión

Para cerrar la sesión actual hay que pulsar el botón “Cerrar sesión” (Ilustración 116), que se encuentra en la esquina superior derecha de la página y pulsar la opción “Aceptar” del mensaje que nos aparecerá por pantalla.

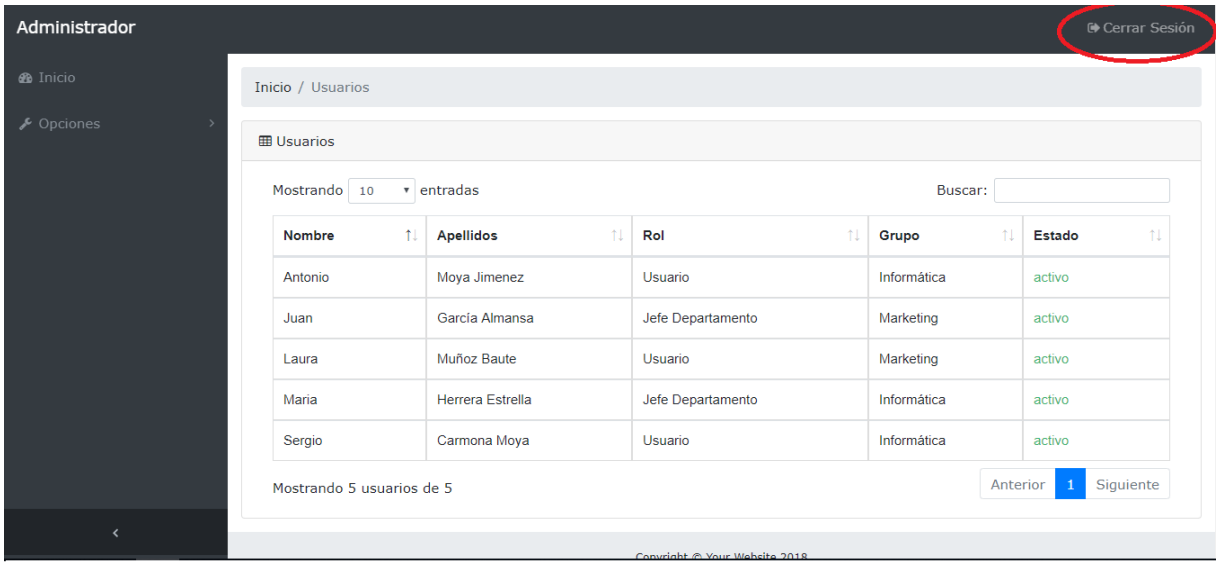


Ilustración 116. Botón Cerrar Sesión.

Anexo III. Contenido CD

1. Memoria en .pdf del trabajo.

2. Código fuente de la página web (Asistencia.rar).

3. Fichero .sql de la base de datos (control.sql).

4. Código del sistema de fichaje (ethernetAsistencia.ino).