

Documentación General

Manual de Cliente

Aeronautic Engines S.A.



Realizada por:



by Adrián, Mario & Tania | Web Developers

García, Tania · Pisabarro, Adrián · Zatón, Adrián
2º DAW

Índice

Análisis de la Idea, Introducción	3
Reto	3
EuskalDev	3
Objetivos del reto	3
Planificación y organización	4
Antes de comenzar	4
Diagrama de Gantt	6
Herramientas utilizadas	6
Wordpress	8
Aplicación web o Intranet	9
Guía de Estilo	9
Guía de Modelo de Control de Versiones	9
Ramas	9
Código JavaScript	10
Clases	10
API Fetch, LocalStorage	11
API Fetch	11
Usuarios	13
Expresiones regulares	13
Otros elementos a tener en cuenta	14
CSS y HTML	14
Ideas que si hubiéramos tenido tiempo nos hubiera gustado implementar	14
Pre-Producción	15

Análisis de la Idea, Introducción

Reto

Desde Aeronautic Engines SA nos piden a EuskalDev la realización de un portal y una página web corporativa.

- **La página web corporativa**
Está desarrollada con el CMS Wordpress en un servidor LAMP de pre-producción. Dispone de los siguientes apartados: “Inicio”, “Blog”, “Conócenos”, “Servicios” y un “Formulario de contacto”.
- **Aplicación interna o Intranet**
Está desarrollada en HTML5, CSS3 y JS.
Enlazada al menú Wordpress nos encontramos con un “login”, “gestión de usuarios” y “gestión de avisos” o como lo hemos llamado “anuncios”.

EuskalDev

EuskalDev ha aceptado el reto para la empresa Aeronautic Engines, esta empresa está formada por los siguientes integrantes:

- Tania García: <https://github.com/TaniaGarciaOlarte>
Funciones: Planificadora y Reportera.
- Adrián Pisabarro: <https://github.com/adrianpisabarrogarcia>
Función: Coordinador
- Mario Zatón: <https://github.com/mariozaton01>
Funciones: Responsable de material y Armonizador.

La empresa EuskalDev se centra en el desarrollo web, realizando así aplicaciones específicas para empresas como AE SA.

Objetivos del reto

Este reto nos ha ayudado a practicar diferentes competencias que en una empresa real nos pueden pedir. Vamos a diferenciar entre los valores como personas que tenemos que tener para afrontar el trabajo en equipo y los objetivos técnicos.

Ante todo, hemos aprendido a ejercer un verdadero trabajo en equipo entre los tres integrantes, a pesar de no haber elegido nosotros los equipos de trabajo hemos trabajado conjuntamente muy a gusto, generando así buen “rollo”, perdona la expresión, entre nosotros. Humildad, trabajo individual, colectivo, empatía y muchas otras competencias hemos conseguido obtener. Si tuviéramos que valorar el trabajo en equipo, no hemos tenido ninguna discusión, ni enfados en el equipo, creemos que hemos trabajado por igual y de forma adecuadas. En ese sentido estamos contentos.

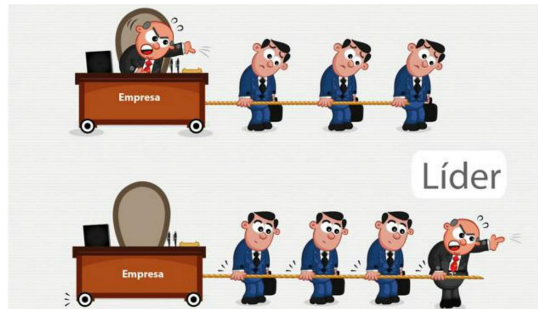


Imagen que demuestra cómo hemos trabajado.

En cuanto a los objetivos técnicos creemos que también los hemos cumplido, a pesar de no haber tenido tiempo físico para poder incrementar en más prestaciones de las que nos pedía el reto, hemos querido dar siempre prioridad a que todos nos turnemos en las diferentes competencias en JavaScript, HTML, CSS, Wordpress y documentación para tratar de hacer todos lo mismo.

Hemos dividido el reto en tres partes prioritariamente; Login, Usuarios y Anuncios/Avisos. Somos 3 personas con 3 tecnologías diferentes las que nos han ayudado a turnarnos y poder adquirir todas las competencias suficientes como para controlar las 3 tecnologías.

Creemos firmemente que se notará a la hora de estudiar para el examen y mejor aún, nos sentimos muy alegres de haber superado con altas expectativas el poder manejar las 3 tecnologías los 3 integrantes, hemos aprendido mucho y nos hemos ayudado mutuamente. Si dos de ellos entendían más de CSS ayudaban a la otra persona cuando le tocaba hacerlo y lo mismo con HTML y JS, todos aportando.

Por ello, nos sentimos satisfechos de seguir esta metodología de trabajo, a pesar de que nos hubiera gustado tener más tiempo para incrementar las funcionalidades de la intranet y estar menos agobiados.

Planificación y organización

Antes de comenzar

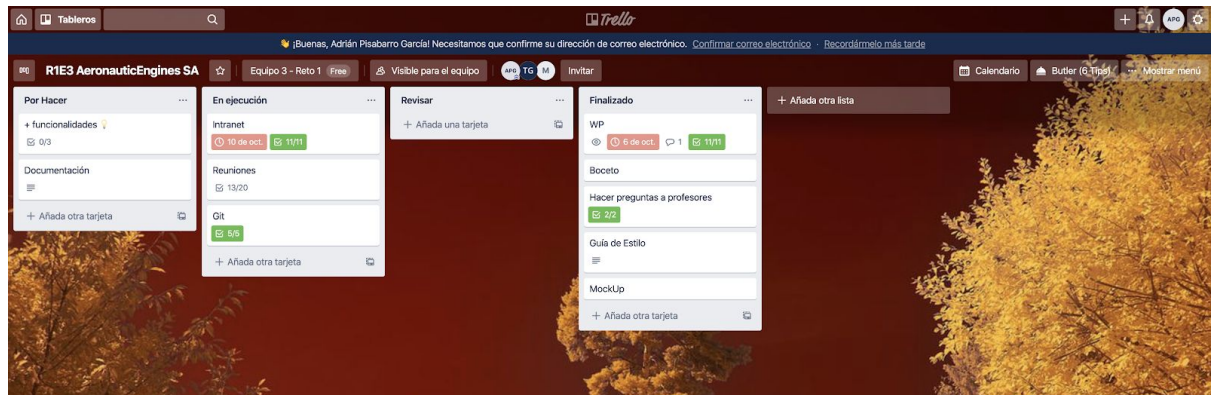
Antes de empezar el reto, nos hemos organizado un día (viernes anterior) antes, en una clase de Jon, de esa manera el lunes siguiente podíamos comenzar con una reunión diaria y ponernos a trabajar directamente.

Hemos realizado reuniones todos los días del reto por la mañana a primera hora respondiendo a las siguientes preguntas siempre:

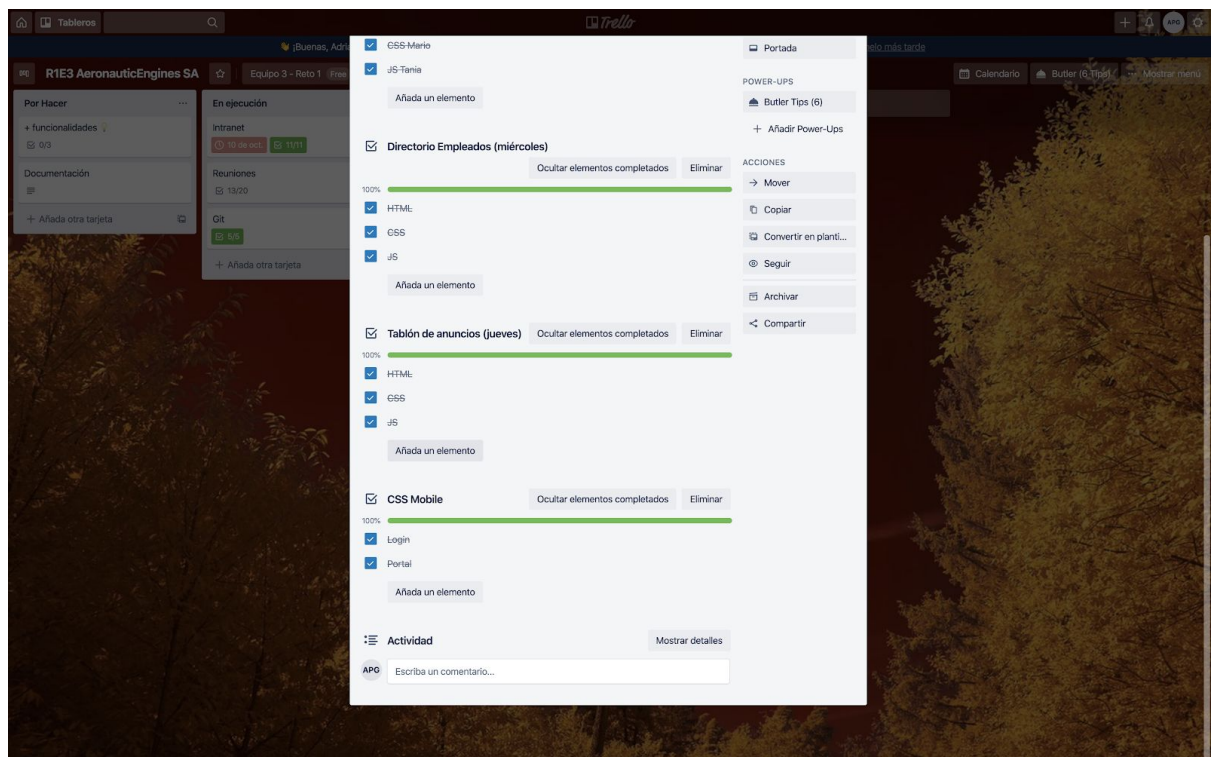
- ¿Qué tal vamos?
- ¿Qué estamos haciendo bien? ¿Qué estamos haciendo mal?
- ¿Deberíamos mejorar algo?
- Tareas para hoy
- ¿Algún problema en el desarrollo del día anterior?

Recordamos que tuvo que hacer un parón Jon para explicar cómo planificarnos, nos sentimos orgullosos porque todo lo que comentaba lo habíamos clasificado por tareas dentro de la aplicación “Trello” para poder planificarnos y ver que íbamos completando el reto.

Como hemos mencionado, dividimos las siguientes tareas, con subtareas:



Tareas



Algunas sub-tareas del apartado kanban de Intranet

Teníamos claro que todos en una de las tecnologías (CSS, Wordpress, HTML, JS) no éramos buenos, lo que nos llevó a pensar que todos teníamos que tocar esa tecnología que no controlábamos bien, para poder aprender y que el resto de compañeros del reto nos ayudasen.

Tuvimos la ayuda de la planificadora (Tania), junto con el responsable de material (Mario) y la coordinación (Adrián) para comenzar con ánimo el reto.

Diagrama de Gantt

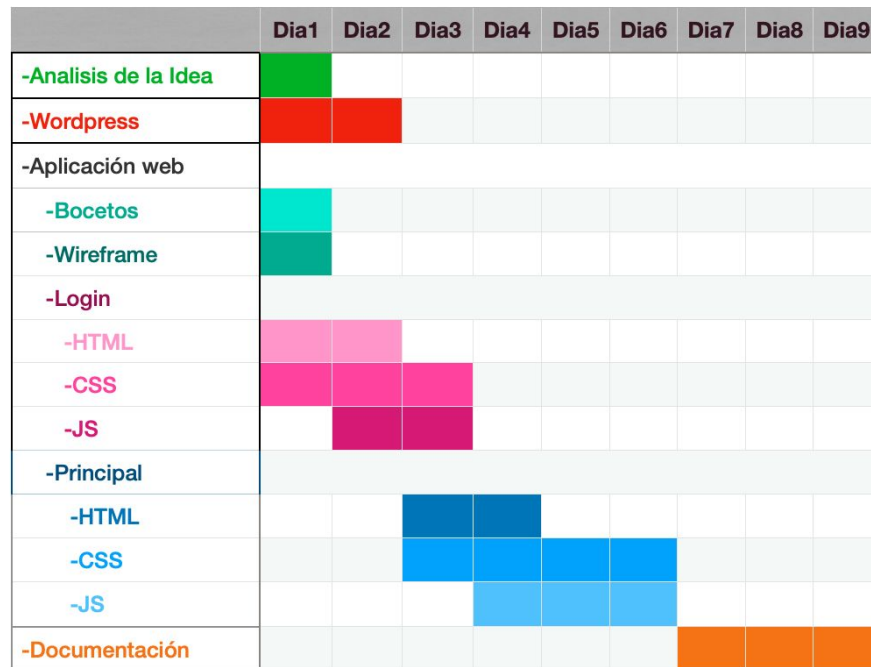


Diagrama de Gantt para planificarnos

Ha sido indispensable este diagrama para poder ver la temporización que podía tener cada tarea que hiciéramos. Aunque en los días 7, 8 y 9 ponga documentación, cabe destacar que estos días también los hemos utilizado para: arreglar errores, tener reuniones para comentar cómo hemos desarrollado cada tarea que hemos hecho para compartir conocimientos, a parte de desarrollar la documentación.

El wordpress no nos llevó mucho tiempo y a la vez se estaban haciendo los bocetos y los wireframes o como nosotros les llamamos MockUps para comenzar el reto, una vez terminadas estas tareas comenzamos con el “meollo” del reto, la intranet, clasificándose en 3 apartados: el login y principal (Usuarios y Anuncios/Avisos). Sub-dividiéndolo en: HTML, CSS y JS.

Herramientas utilizadas



Estas han sido algunas de las herramientas que hemos utilizado, sin embargo, pasamos a mencionar cuáles han sido.

- Trello
Nos ha ayudado a visualizar las tareas y a ir completando nuestro tablero kanban
- GitHub
A pesar de haber encontrado problemas por no haber realizado anteriormente ninguna práctica en grupo con GitHub, no ha ido tan mal como los problemas que nos hemos ido encontrando.
No conocíamos exactamente el poder del merge y el pull request para comprar código (borrado y añadido). Teníamos miedo de que nuestro código se solapara o no quedara como nosotros queríamos a la hora de pasarlo de una rama a otra y unirlo. Es por ello que hemos utilizado para acompañar de nuestro repositorio Git: Google Drive (donde hacíamos BackUps diarios) y WinMerge (que nos ha ayudado a unir diferentes archivos de código, una herramienta muy fácil de usar y recomendada para grupos de trabajo entre programadores y unir código) aunque gracias a esta última hemos comprendido el funcionamiento de Git.

En todo momento hemos utilizado la consola y terminal para subir nuestros proyectos, en ningún momento otro software como GitHub desktop, Git Kraken o la interfaz de los programas de JetBrains.

- JetBrains: PhpStorm y WebStorm
Han sido los principales IDEs para poder trabajar con la aplicación. Hemos tenido que añadir un archivo .gitignore a nuestro repo para poder excluir los archivos ocultos que generan estas aplicaciones, sin embargo, estamos encantados con estos editores de código.
- Notion
Como bloc de notas para el reto
- Discord
Nuestro canal de comunicación principal, para compartir reuniones y comunicaciones del equipo (es parecido al Slack). Fuera de nuestro “horario de trabajo”, las comunicaciones han ido vía WhatsApp con el grupo que tenemos creado.
- Google Suit for Education
Proporcionado por Egibide. Herramienta para compartir archivos y editarlos online, como este documento.
- Trello
Nos ha ayudado a visualizar las tareas y a ir completando nuestro tablero kanban
- LAMP y XAMP
Para la realización del Wordpress y la base de datos.

- Wordpress CMS
CMS utilizado para crear la página web de AE SA.
- Netlify
Para pasar a pre-producción y generar una vista de una rama del repositorio Git.



Tablero kanban

Wordpress

Se puede acceder mediante el ordenador 6 de la aula (2GDAW06) y una vez encendido podemos buscar en el cmd que IP esta utilizando para poder visualizar la web en cualquier ordenador de la misma red o rub-red.

La plantilla de wordpress es una plantilla premium que conocíamos y que nos ha facilitado el trabajo. Además, hemos utilizado nuestros conocimientos en CSS para cambiar elementos de la plantilla que no nos convencían.

Hemos decidido juntar todas las páginas y crear una “Landing Page” para apoyar el uso del scroll. Hoy en día utilizamos bastante los móviles, las webs mayoritariamente los usuarios las consultan con dispositivos como tablets y móviles que nos acostumbran a hacer scroll, esa ha sido la razón de en vez de utilizar diferentes páginas, utilizar una única página con todo el contenido que nos pedían.

En algunos casos el texto está extraído de la web de Aernova y en otras ocasiones un simple Lorem Ipsum.

Infografías extraídas de “Pexels” y “Freepik”.

Aplicación web o Intranet

Guía de Estilo

Nuestra guía de estilo es la siguiente, se encuentra en el ANEXO 1 de la documentación entregada:

https://drive.google.com/file/d/15FvCuQwqUQ6f_ciCLILKsc3CC1yf7IRQ/view?usp=sharing

Guía de Modelo de Control de Versiones

Nuestra guía de funcionamiento con Git es la siguiente, se encuentra en el ANEXO 2 de la documentación entregada:

<https://drive.google.com/file/d/1moQMAYleIT9nnNBsAmhj6PIOFs80t9ZE/view?usp=sharing>

A modo de resumen:

Ramas





Master: Rama para las versiones definitivas

Hotfix: Rama de pequeños cambios del master

Develop: Rama secundaria para almacenar los cambios diarios. De la rama develop salen las siguientes ramas:

- **CSS:** A partir del primer guardado del archivo hemos utilizado esta rama para trabajar con el CSS
- **JS:** En esta rama se trabaja en el JS a partir de una versión avanzada del proyecto para evitar problemas
- **HTML:** Rama en la que solo se hemos trabajado el HTML

*Hemos tenido algún conflicto en algunas ramas por que las carpetas JS, CSS tenían el mismo nombre como las ramas, lo cual en algún momento hemos tenido que crear otras ramas de trabajo con diferentes nombres, pero que cumplían la función que se describe.

All branches			
master	Updated 1 hour ago by adrianpisabarrogarcia	Default	
develop	Updated 10 hours ago by adrianpisabarrogarcia	2 8	New pull request 
CSS	Updated 2 days ago by mariozaton01	2 1	New pull request 
JS	Updated 3 days ago by TaniaGarciaOlarte	2 8	New pull request 
HTML	Updated 3 days ago by mariozaton01	2 0	New pull request 

Enlace del repositorio github:

<https://github.com/adrianpisabarrogarcia/aeronautic-engines.git>

Código JavaScript

Clases

Hemos creado dos clases diferentes, una la de Usuarios donde guardaremos todos los datos de los usuarios de los empleados y otra clase de Avisos/Anuncios donde guardaremos toda la información de los avisos que añadan los usuarios.

Los atributos de las clases son:

Usuarios:

- Nombre
- Apellido
- DNI
- Usuario
- Contraseña



```
objetousuario.js
class Usuario{
  constructor(usu,con,nombre,apellido,dni) {
    this.usuario = usu;
    this.contra = con;
    this.nombre = nombre;
    this.apellido = apellido;
    this.dni = dni;
  }
}
```

Imagen de ejemplo de la clase Usuarios.

Anuncios:

- Título
- Fecha
- Descripción



```
class Aviso{
  constructor(titul, fech, descrip) {
    this.titulo= titul;
    this.fecha= fech;
    this.descripcion= descrip;
  }
}
```

Imagen de ejemplo de la Clase Anuncios

API Fetch, LocalStorage

A pesar de que hemos utilizado la tecnología de LocalStorage para el almacenamiento de datos, disponemos la de la API Fetch para el backend de nuestra web incorporada en comentarios.

La verdad, es que nos ha sido muy simple la utilización del LocalStorage y podíamos acceder con total facilidad a los datos.

API Fetch

Añadir

Para añadir los datos al servidor utilizaremos el método POST e indicaremos qué datos queremos introducir en el servidor con JSON.stringify

```
fetch("url.php", {  
  method: 'POST',  
  body: JSON.stringify(datos),  
  headers: {  
    'Content-Type': 'application/json'  
  }  
}).catch(error => console.log('Error:', error));
```

```
/*fetch("url.php", {  
  method: 'POST',  
  body: JSON.stringify(listaAnuncios),  
  headers: {  
    'Content-Type': 'application/json'  
  }  
}).catch(error => console.log('Error:', error));*/  
}
```

Modificar

Para modificar los datos en el servidor utilizaremos el método PUT y al igual que para añadir los datos debemos indicar los datos que queremos añadir

```
fetch("url.php", {  
  method: 'PUT',  
  body: JSON.stringify(datos[x]),  
  headers:{  
    'Content-Type': 'application/json'  
  }  
}).catch(error => console.log('Error:', error));
```

```

/*fetch("url.php", {
  method: 'PUT',
  body: JSON.stringify(datosUsu),
  headers:{
    'Content-Type': 'application/json'
  }
}).catch(error => console.log('Error:', error));*/

```

Borrar

Para eliminar los datos en el servidor utilizaremos el método DELETE e indicamos el dato que deseamos eliminar

```

fetch("url.php", {
  method: 'DELETE',
  body: JSON.stringify(datos[x]),
  headers:{
    'Content-Type': 'application/json'
  }
}).catch(error => console.log('Error:', error));

```

```

/*fetch("url.php", {
  method: 'DELETE',
  body: JSON.stringify(datosConectado),
  headers: {
    'Content-Type': 'application/json'
  }
}).catch(error => console.log('Error:', error));*/

```

Consultar datos

Para consultar los datos utilizaremos el elemento donde queremos mostrar los datos, los datos que queremos mostrar y el método de response

```

fetch('url.json')
  .then(response => response.json())
  .then(data => data.forEach(elemento =>
    document.getElementById("todosFichero").value
    += elemento.nombre + " " + elemento.edad + "\n"));

```

```

/*fetch('url.json')
  .then(response => response.json())
  .then(data => data.forEach(elemento => document.getElementById("listaUsuarios").innerText += elemento.nombre + " " + elemento.apellido + " " +
  elemento.dni + " " + elemento.user + "\n"));*/

```

Guardar los datos

Para guardar los datos del servidor en un array debemos indicar la url del servidor, el método response, donde queremos guardar los datos y que queremos guardar

```

fetch (url servidor){
  then ((response) => {
    return response.json();
  })
  then ((datos) => {
    listaAviso = JSON.stringify(datosAviso);
  });
};

```

```

/*fetch(url servidor){
  then((response) => {
    return response.json();
  })
  then((datos) => {
    listaAviso = JSON.stringify(datosAviso);
  });*/

```

LocalStorage

Añadir

Para añadir en el localStorage debemos utilizar el método setItem en el cual indicamos donde queremos que se guarde y lo que queremos que se guarde

```

localStorage.setItem('datos',JSON.stringify(usuarios));

```

Borrar

Para borrar lo primero que realizaremos será borrar los datos del array con el método splice y luego añadirlos al LocalStorage con el método setItem

```

datosUsu.splice(i,1)
localStorage.setItem('datos',JSON.stringify(datosUsu));

```

Modificar

Para modificar lo primero que deberemos hacer es realizar la modificación en el array y luego añadirlos al LocalStorage con el método SetItem

```

datosUsu[i].nombre = document.getElementById( elementId: "name").value
datosUsu[i].apellido = document.getElementById( elementId: "surname").value
datosUsu[i].dni = document.getElementById( elementId: "credencial").value
datosUsu[i].usuario = document.getElementById( elementId: "buscar_usuario").value
datosUsu[i].contra = document.getElementById( elementId: "pass").value

localStorage.setItem('datos',JSON.stringify(datosUsu));

```

Guardar los datos

Para guardar los datos del localStorage utilizaremos el método .parse

```
let datosusu = JSON.parse(localStorage.getItem( key: "datos"));
```

Usuarios

No hemos querido hacer una distinción entre usuarios empleados y usuarios administradores, con diferentes derechos. Leyendo bien el enunciado del proyecto, no nos lo pedían expresamente y hemos priorizado que todos los requisitos estén antes de hacer distinciones. Sin embargo, veríamos bien llevarlo a producción, con tiempo. Es algo que nos gustaría haber tenido tiempo para implementarlo.

Expresiones regulares

Usuarios

- usuario, contraseña: caracteres alfanuméricos
- nombre, apellido: caracteres del alfabeto y blancos
- DNI: comprobación de un DNI real (solo se almacenará en tu ordenador a si que se puede probar)

Avisos/Anuncios

- título: caracteres del alfabeto y blancos
- fecha: formato dd/mm/yyyy fecha anterior a la de hoy.
- descripción cualquier carácter.

Otros elementos a tener en cuenta

A modo de referencia hemos utilizado mayoritariamente estas páginas webs <https://developer.mozilla.org/es/> y <https://es.stackoverflow.com> para dudas que anteriormente a otros usuarios les habían surgido.

Además, nunca hemos implementado/copiado código que no entendíamos. Siempre lo hemos probado y jugado con él y hemos entendido el funcionamiento del mismo antes de implementarlo.

CSS y HTML

Tenemos la suerte que el curso pasado dimos a fondo los lenguajes de marcados y de cascada para poder utilizarlos en este reto.

Nuestro portal está adaptado a móvil y tablet. Además hemos tratado también de que en la orientación: landscape también se visualizará correctamente. Diseño “responsive”.

Creemos que es una aplicación sencilla, visual y adaptada a la accesibilidad de cualquier usuario, siempre hemos pensado en nuestros padres, que no tienen mucha idea, al fin y al cabo serían los usuarios modelo que se podrían encontrar la aplicación en su trabajo.

Ideas que si hubiéramos tenido tiempo nos hubiera gustado implementar

Creemos que hemos cumplido con las funcionalidades que nos pedían en el reto.

Propuestas para mejorar en un futuro de cara a la producción:

- Incluir un evento de “intro” para los campos y formularios
- Hacer la distinción entre un empleado normal y un administrador (roles)
- Adaptarlo mayormente a una orientación: landscape
- Añadir efectos css
- Realizar tablas dinámicas para gestionar mejor la información y resumir el CRUD en una sola tabla y no hacer uso de tener que saber el nombre del usuario o el nombre del anuncio/aviso.

Hemos encontrado dificultades en el control de versiones que finalmente sabemos manejar, cada uno controlabamos tecnologías diferentes con lo cual no hemos tenido tiempo de innovar todo lo que nos gustaría y en ocasiones hemos tenido que preguntar e investigar demasiado, ayudado de otro compañero del reto.

Bibliografía

Agradecemos toda documentación y ayuda que nos ha prestado:

- Apuntes de Ikas: <https://ikas.egibide.org>
- Ayuda de diferentes profesores que han probado la aplicación
- <https://stackoverflow.com>
- <https://developer.mozilla.org/es/>
- <https://www.w3schools.com>
- Apuntes de Jaime y Jon de CSS del curso pasado
- Ejercicios resueltos de Nieves de JS

Pre-Producción

Te dejamos los enlaces relevantes de este proyecto para echarle un vistazo:

- Github: <https://github.com/adrianpisabarrogarcia/aeronautic-engines>
Dispondrás de la rama master con todo el proyecto el mismo día de la presentación
- Wordpress: (hace falta tener encendido el ordenador 6 del aula con el so w10)
<http://172.20.224.111/ae/> ó <http://localhost:8080> si queremos visualizarlo solo desde el mismo PC nº6 del aula.

- Intranet: <https://aeronauticengines.netlify.app>