

CFGS DAW

Desarrollo de Interfaces Web

Memoria del mockup de un proyecto web de una consultora tecnológica de proyectos informáticos y de la codificación de su landing-page

Jorge Muñoz Carrión
Santiago Sánchez Sans

Fecha de entrega: 18 noviembre de 2024

Índice general

1. Introducción	2
1.1. Justificación	2
1.2. Objetivos	2
1.2.1. Características de la aplicación	2
1.3. Comparativa	3
2. Análisis y Planificación	12
2.1. Análisis	12
2.1.1. Requisitos funcionales de los usuarios	12
2.1.2. Requisitos de la aplicación	14
2.1.3. Lenguaje en la parte cliente	17
2.2. Planificación y fases del desarrollo	19
2.2.1. Fase 1*: Uso de Trello para distribuir tareas	19
2.2.2. Fase 2*: mockup con figma	20
2.2.3. Fase 3*: GitHub y desarrollo colaborativo	22
3. Desarrollo	27
3.1. Diseño	27
3.2. Funcionalidad	30
3.3. Codificación	30
3.4. Mantenimiento	38
4. Ampliaciones	40
4.1. Mejora de código	40
4.2. Soporte de idiomas	40
5. Evaluación y Conclusiones Finales	41
6. ANEXO: memoria con Figma	42
6.1. Landing Page (index.html) con dispositivos	43
6.2. Landing Page (index.html) sin dispositivos	44
6.3. Front-end (página interna)	45
6.4. Back-end (página interna)	46
6.5. IA y BigData (página interna)	47
6.6. Incorpórate	48
6.7. Quiénes Somos	49
6.8. Contacto	50
7. Bibliografía	50

Introducción

1.1. Justificación

Hemos pensado que como proyecto de la asignatura “desarrollo de interfaces web” podríamos desarrollar una página web visualmente atractiva para una consultoría tecnológica que ofreciese sus hipotéticos servicios. Debería ser una web que consiguiese atraer a potenciales clientes del sector tecnológico dispuestos a contratar servicios de desarrollo informático en distintas áreas.

Quizás es un objetivo altamente ambicioso o incluso irrealista, en tanto que se encuentra ante restricciones de tiempo, acotado a los tempos académicos del grado superior. Sin embargo, perseguirlo nos ayuda a crear un producto que esperemos al menos sea agradable a la vista e invite a conocer más a los desarrolladores que hay detrás.

1.2. Objetivos

Esta hipotética consultora ofrecería servicios front-end, back-end y de IA y Big Data. La finalidad de la web debería ser, pues, ofrecer una experiencia de usuario altamente atractiva, sencilla, en línea con las tendencias actuales del mercado front-end (uso de gradientes lineales, contenedores no solamente rectangulares sino de formas curvas o torcidas y elementos interactivos al pasar por encima con el ratón) que mostrase a su vez un mínimo de capacidades de los desarrolladores front-end de la supuesta consultoría tecnológica, que habrían sabido adaptarse a las tendencias actuales de la industria para hacer un producto llamativo. Debería ser, en sí misma, una declaración de intenciones, de modo que no solamente fuese la forma de darse a conocer de la empresa que hay detrás sino también la forma de enseñar lo que sus empleados dedicados al cliente web son capaces de hacer.

1.2.1. Características de la aplicación

En esta memoria se puede observar el mockup completo, hecho con Figma, en el anexo. Se invita al lector a dar una ojeada por encima al anexo (ver capítulo 6 y luego seguir leyendo desde aquí).

En esencia, el diseño de la web debe ser minimalista, con poco texto y muy orientado a la parte gráfica. Con facilidades para acceder a consultar los distintos servicios que la consultora tecnológica ofrezca (en este caso en tres ámbitos: front-end, back-end, IA y BigData), facilidades para contactar con su equipo de desarrolladores y para ver quienes están detrás de la empresa. Todo ello debe incluirse en la barra de navegación. En esta asignatura no se podrá usar JavaScript, pero sí se permitirá el uso de ciertas librerías que hagan uso de este lenguaje. Para más información ver el apartado [Requisitos de la aplicación](#).

1.3. Comparativa

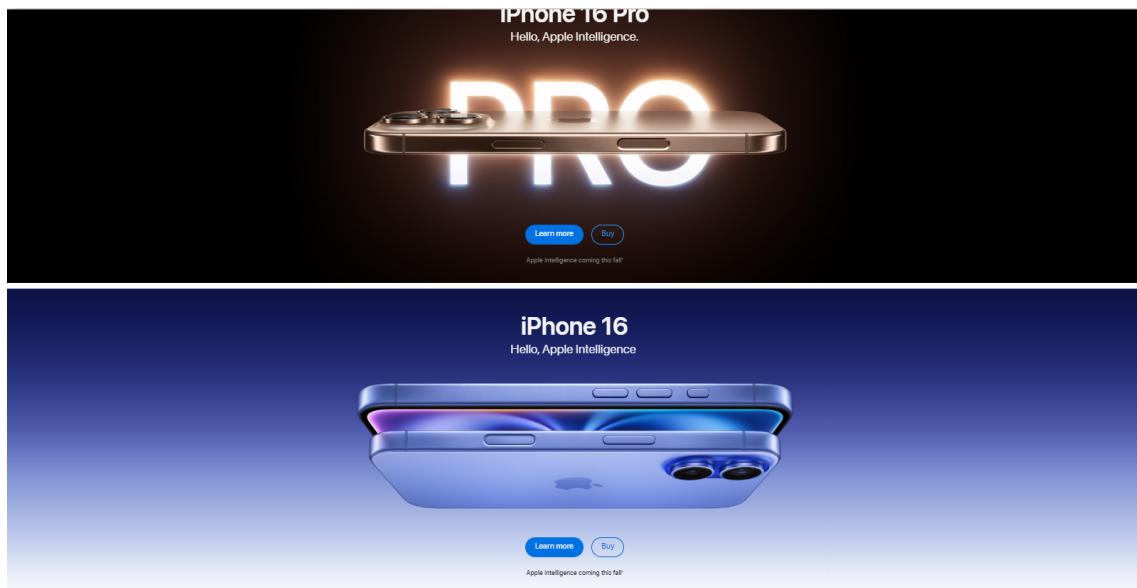
Para realizar esta web nos hemos inspirado en otras webs. Aquello que funciona debe emularse, no hay que reinventar la rueda. En toda creación artística (desde la música, pasando por el cine, hasta el mock-up de una página web) se debe utilizar la parte justa de repetición para no agobiar al usuario con novedades que no puede procesar.

Si utilizamos algo excesivamente distinto a las tendencias actuales, la gente lo rechazará por ser poco familiar, extraño (en música y cine por no saber apreciarlo, en desarrollo web por no encontrar el botón de contacto). Si utilizamos lo mismo que usan los grandes jugadores de la industria, pareceremos más profesionales. Eso explica por qué en las películas el argumento se repite una y otra vez, o por qué en la música escuchamos ritmos muy parecidos y transiciones similares. El ser humano necesita un cierto sentimiento de familiaridad y los creadores artísticos lo saben y generan sus nuevos contenidos tomando prestados elementos ya usados cuya combinación ha resultado ser apreciada en el pasado.

Aquí hemos hecho lo mismo. Por poner un ejemplo, para la landing page se han usado ideas que funcionan y que otras empresas ya implementan.

Por ejemplo, nos ha gustado el doble gradiente lineal y vertical que se utiliza en la web de **Apple** (figura 1.1). Para promocionar su iPhone el gigante de la manzana utiliza un contendor negro, y debajo del mismo tenemos un fondo de transición de color azul oscuro a azul claro, y de azul claro a blanco (lo definimos, pues, como una especie de doble gradiente lineal):

Figura 1.1: Uso de gradiente lineal en la web de apple



Nosotros en nuestra landing page hemos optado por hacer algo parecido, pero a la inversa (figura 1.2). En la sección front-end pasamos de color blanco a azul, mediante un primer gradiente lineal vertical; y luego pasamos de azul a azul oscuro, mediante un segundo gradiente lineal (transicionando ese azul oscuro luego en la sección back-end, que será negra):

Figura 1.2: Uso de doble gradiente lineal en nuestra landing page.



Luego, para hacer el contenedor en forma de onda de la landing page, nos hemos inspirado en la página web de **Trello**. Nos ha gustado la combinación de colores y el gradiente lineal de izquierda a derecha:

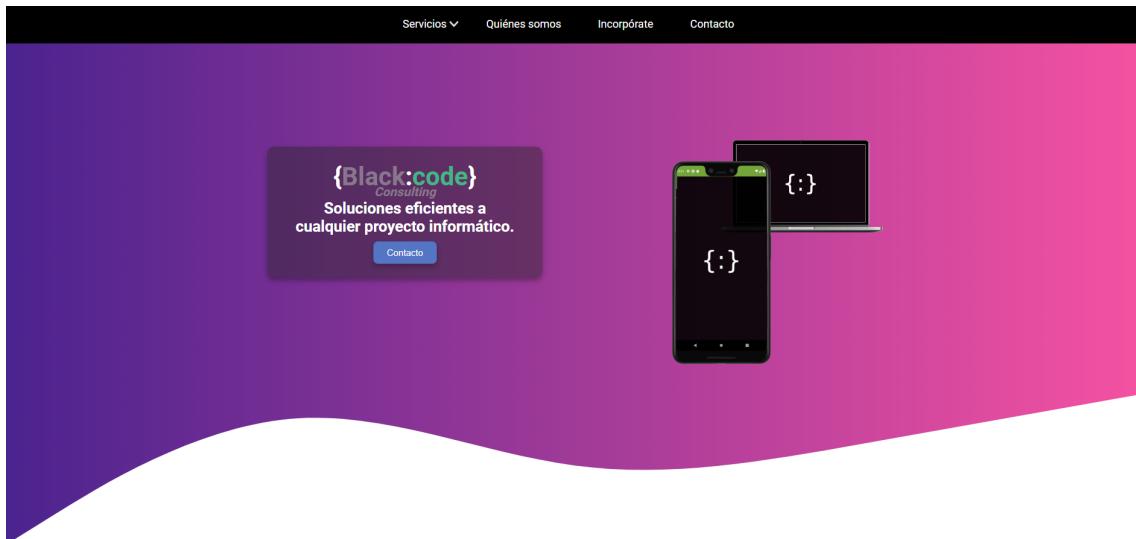
Figura 1.3: Uso de gradiente lineal horizontal en la web de Trello.



Su estética la hemos emulado en la sección del contenedor que presenta la em-

presa, tal que así:

Figura 1.4: Uso de gradiente lineal horizontal en nuestra web.



Para la barra de navegación nos ha gustado la simplicidad y estilo de la página web de **Accenture** con pocos elementos en la barra de navegación, tipografía en blanco y fondo negro:

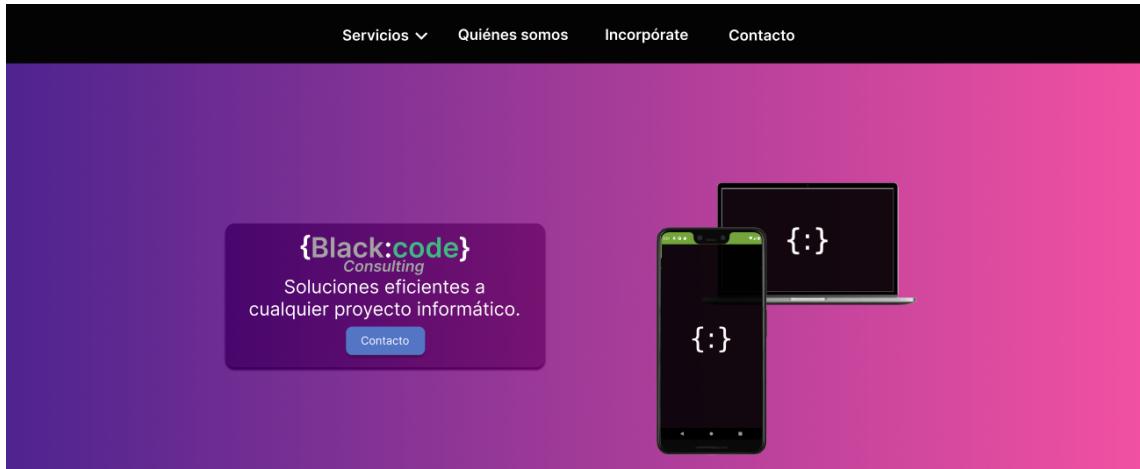
Figura 1.5: Barra navegación de Accenture (fondo negro y letra blanca)



Esta estética nosotros la hemos incorporado al diseño de nuestra “NavBar”¹, que en este caso solo tendrá un solo *menú desplegable* y un elemento más para albergar una pestaña extra, la pestaña de contacto:

¹ Esta estética también ha sido aplicada, en cierta medida, en nuestra footer; para conseguir consistencia con la barra de navegación.

Figura 1.6: nuestra “NavBar”: fondo negro, letra blanca y un solo desplegable.



Luego, para la sección de contacto hemos tomado prestada la idea de la sección de contacto de **NextPort.ai** con un gradiente horizontal y con un *formulario* en la parte derecha con solo los campos imprescindibles para permitir el contacto:

Figura 1.7: Página de contacto de NextPort.ai

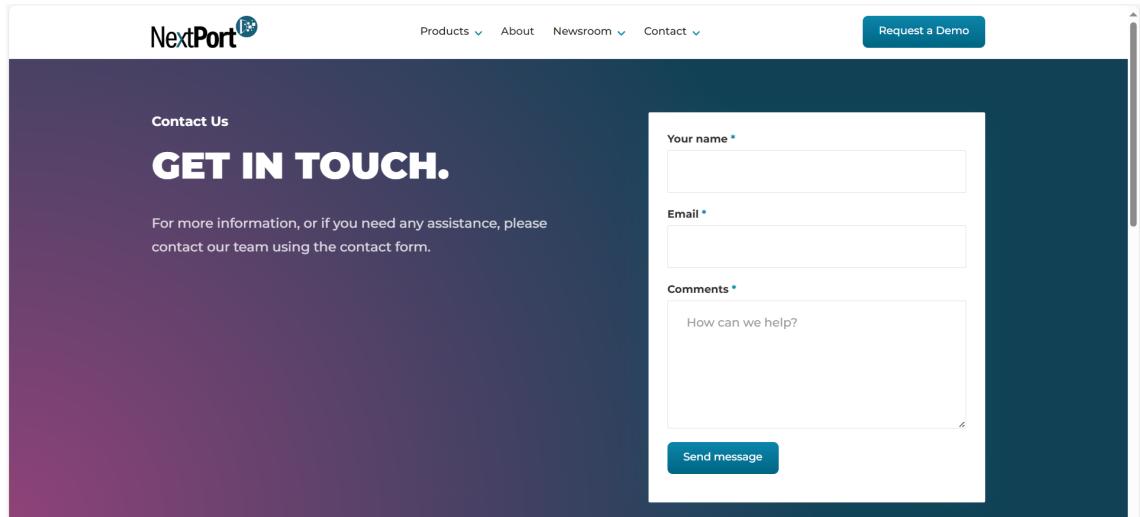


Figura 1.8: Nuestra página de contacto, con un *formulario*.

The screenshot shows a contact form integrated into a website. At the top, there is a navigation bar with links: 'Servicios ▾', 'Quiénes somos', 'Incorpórate', and 'Contacto'. Below the navigation, the main content area has a dark purple gradient background. On the left, there is a large heading 'Contáctanos' and a text block: 'Para obtener más información, contratar nuestros servicios o preguntarnos dudas sobre cómo podemos ayudarte a realizar tu proyecto, contáctanos utilizando el formulario de contacto.' To the right of this text is a white rectangular form box containing three input fields: 'Tu nombre*', 'E-mail*', and 'Comentarios*'. Below these fields is a blue 'Contacto' button.

Para hacer el apartado web de *trabaja con nosotros* nos hemos inspirado en la web de **soprasteria.es** ([link](#)). Hemos usado la misma estructura, una imagen parecida de equipo joven y una descripción inspiradora para así transmitir unos valores corporativos sólidos al usuario y poder llegar al objetivo de captar talento para la empresa (ver la figura 1.9 de la web, y comparar con parte de un fragmento de nuestra página en la figura 1.10):

Figura 1.9: Sección para captar talento de la web de Soprasteria. Con una combinación de colores sencilla pero eficiente.

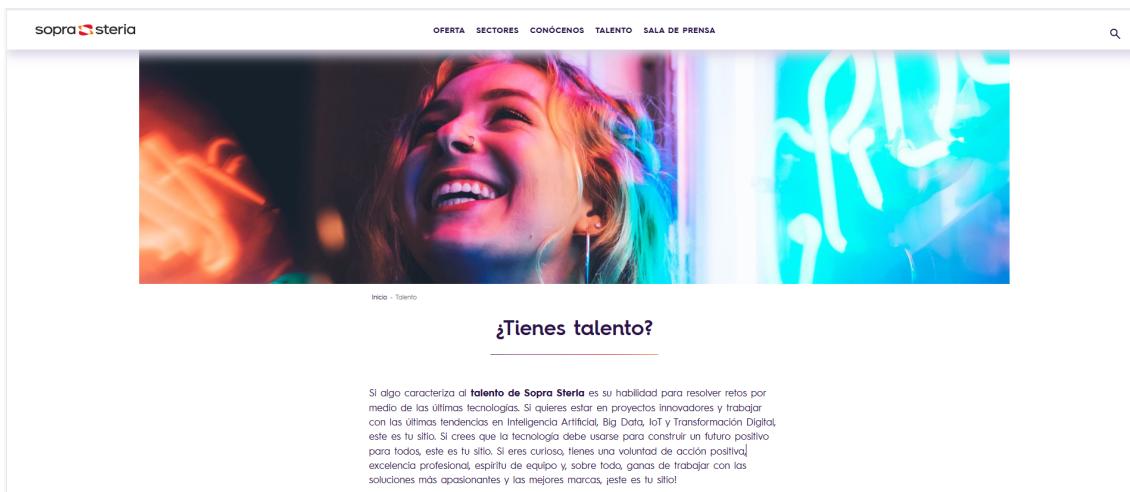


Figura 1.10: Un fragmento de la sección para captar talento de nuestra web, a la que llegamos clicando “incorpórate” en la navBar. Fue inspirada en la sección equivalente de la web de Soprasteria. Se puede ver completa en el anexo.



Finalmente, comentar que para el apartado “quienes somos” de nuestra web nos hemos inspirado en la *landing page* de **hubspot.es** ([link](#)). Hemos usado la misma estructura de tarjetas, y texto. Para dar a conocer nuestra empresa elegimos el diseño simple, minimalista y con colores claros para transmitir al usuario final transparencia, claridad y un fácil acceso a la información (véase la web de hubspot en la figura 1.11 y comparadlo con la nuestra en la figura 1.12). La diferencia es que nosotros hemos incorporado en la parte superior de la página, justo debajo de la navBar, una *galería de vídeo* en lugar de una imagen fija. Además, también hemos incorporado *una ubicación* en la parte inferior de la página, justo por encima del footer pero separada de ésta:

Figura 1.11: Página web de Hubscop, sobre la que nos hemos inspirados para hacer la sección “quienes somos” de nuestra web.

The screenshot shows the 'About Us' section of the HubSpot website. At the top, there's a navigation bar with the HubSpot logo, menu items like 'Productos', 'Soluciones', 'Precios', 'Recursos', and a red button 'Obtener HubSpot gratis'. Below the navigation, the title 'Sobre nuestra empresa' is displayed in a large, bold, dark font. A paragraph of text follows, stating: 'En HubSpot, nuestra empresa y nuestra cultura se parecen mucho a nuestro producto. Están conectadas, no ensambladas, para lograr una excelente experiencia.' To the right of the text is a group photo of the HubSpot team sitting on a balcony with a cityscape and mountains in the background.



Nuestra misión: Ayudar a millones de empresas a crecer mejor

Creemos que no sólo es importante crecer más, sino crecer mejor. Y esto significa garantizar el éxito de tu empresa y también el de tus clientes. Todo el mundo gana.

Nuestra historia

En 2004, cuando eran compañeros de posgrado en el MIT, Brian Halligan y Dharmesh Shah se dieron cuenta de que la manera de comprar estaba cambiando. La gente no quería estar recibiendo publicidad constantemente, quería información útil. En 2006, fundaron HubSpot para ayudar a las empresas a aprovechar ese cambio para crecer mejor con el inbound marketing.

Desde entonces, HubSpot ha ido más allá del marketing y se ha convertido en una plataforma conectada, no ensamblada, que permite crear la experiencia del cliente que la gente quiere. HubSpot, dirigida por la directora ejecutiva Yamini Rangan, utiliza su plataforma de clientes, desarrollada en integración con el Smart CRM con tecnología de inteligencia artificial, para ayudar a millones



Figura 1.12: Fragmento de la página interna para dar a conocer la empresa, a la que enlazamos al hacer click sobre el link “quienes somos” de la navBar. Versión completa en anexo: apartado [6.7](#).



The screenshot shows a website header with navigation links: "Servicios", "Quiénes somos" (highlighted in blue), "Incorpórate", and "Contacto". Below the header is a large photograph of four people (three men and one woman) working together around a table in an office environment. A play button icon is overlaid on the photo. The main title "Sobre nuestra empresa" is centered below the image.



Nuestra Historia:

En Black-Code Consulting, comenzamos con una visión clara: crear soluciones tecnológicas que marcaran la diferencia en un mundo digital en constante evolución.

Desde nuestros primeros proyectos, hemos crecido junto a nuestros clientes, desarrollando herramientas y plataformas innovadoras que impulsan sus negocios.

En Black-Code Consulting, nuestra empresa y nuestra cultura se parecen mucho a nuestro producto. Están conectadas, no ensambladas, para lograr una excelente experiencia.

Nuestra misión: Ayudar a millones de empresas a crecer mejor

Creemos que no sólo es importante crecer más, sino crecer mejor. Y esto significa garantizar el éxito de tu empresa y también el de tus clientes. Todo el mundo gana.



Análisis y Planificación

2.1. Análisis

2.1.1. Requisitos funcionales de los usuarios

Debemos preguntarnos **qué servicios** ofrece la página. La página web tiene la finalidad de proporcionar tres servicios propios de la empresa: desarrollo en tecnologías front-end, desarrollo en tecnologías back-end y desarrollo en el mundo del dato e Inteligencia Artificial.

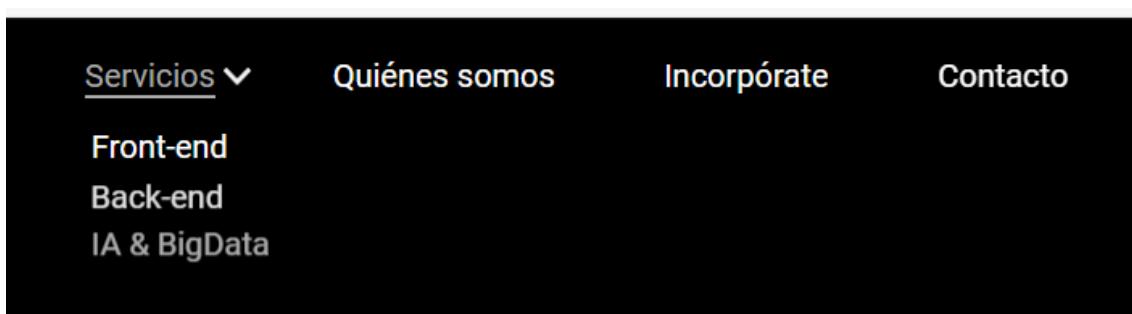
A tal efecto va a tener una landing page (index.html) que va a mostrar los tres servicios proporcionados de forma esquematizada. Si se accede a la barra de navegación o a la footer se podrá llegar a los demás servicios mediante hipervínculos.

Figura 2.1: Barra de navegación con sus cuatro elementos.



Dentro de la sección servicios hay un desplegable o “drop-down” que nos permite llegar a las distintas páginas internas: la página interna de front-end (véase 6.3), la página interna de back-end (véase sección 6.4) y la página interna de IA y BigData (véase sección 6.5).

Figura 2.2: Barra de navegación con el desplegable activo.



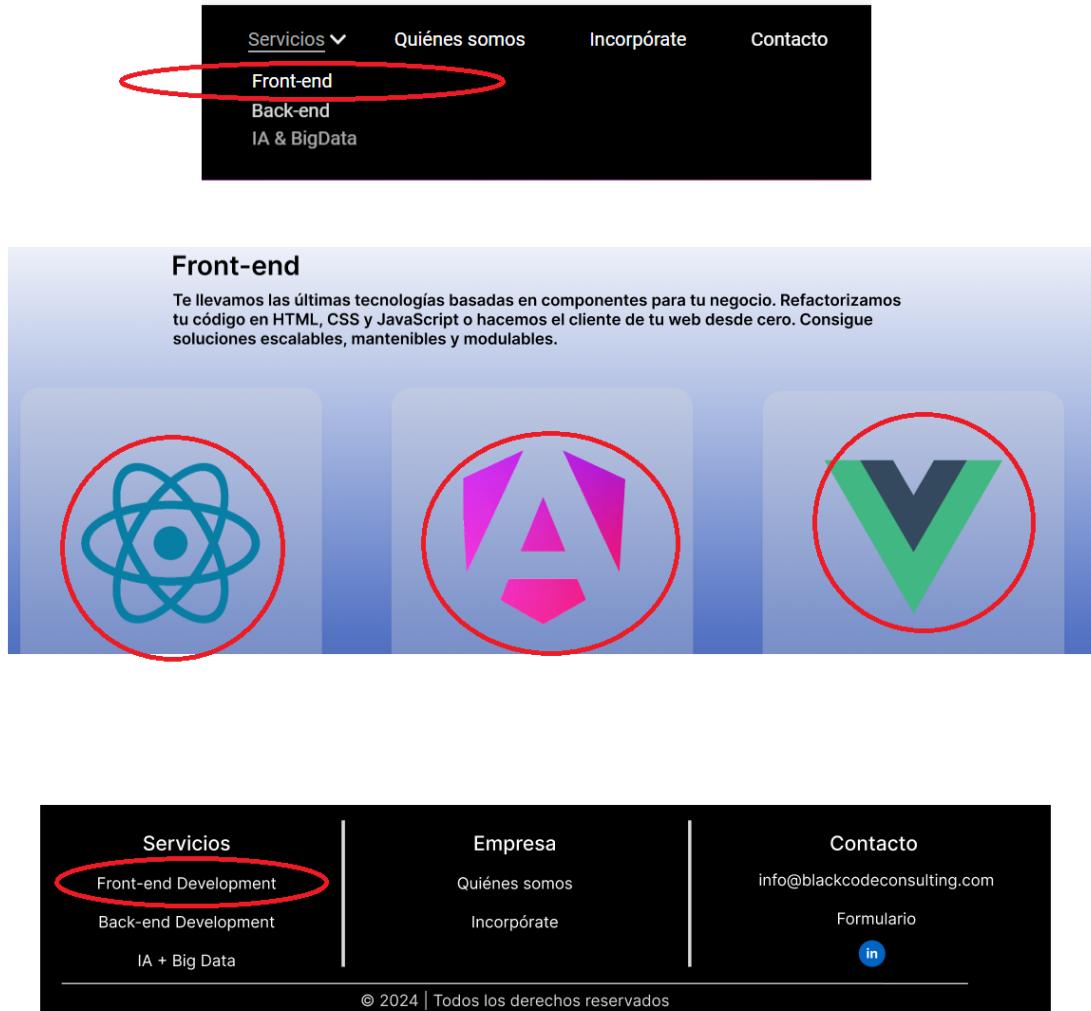
Desde esta misma barra es posible acceder a las secciones que presentan a los empleados de la empresa (Quiénes somos, ver sección 6.7), la sección de adquisición

del talento para conseguir que candidatos manden su currículum (Incorpórate, ver sección 6.6) y la sección de contacto (sección 6.8).

Además **todos los links** de esta barra de navegación también se encuentran en la **footer**. Esta **redundancia** que tiene la página web hace que la información empresarial le sea accesible y el contacto, fácil. Los usuarios deberían interaccionar con la página haciendo un “scroll” rápido por la landing page. Si lo hacen encontrarán que en cada imagen de una tecnología hay un link que se asocia con una subpágina interna de la web, siendo este link justamente el mismo link al que nos redirigirá cualquiera de los hipervínculos de la pestaña servicios de la *navBar* o de la **footer**. Esta redundancia no es casual, pues queremos que se pueda llegar al mismo punto de maneras parecidas, como vemos en la figura 2.3.

En esencia **los requerimientos del cliente** que entre en la web serán simplemente que la web sea usable. Acceso rápido, fácil e intuitivo a las distintas páginas internas que conforman el proyecto web.

Figura 2.3: Ejemplo de acceso de forma redundante a la subpágina de front-end, tanto por menú, como por footer, como por imágenes en la sección de tecnologías front-end de la landing page. La redundancia construye accesibilidad para el usuario.



2.1.2. Requisitos de la aplicación

El **primer requisito** que tenemos para este proyecto es que no se puede utilizar JavaScript. Son restricciones de la asignatura dado que las competencias de la misma versan sobre el desarrollo de interfaces de usuario, no sobre el desarrollo cliente o front-end en su totalidad.

El **segundo requisito** es cumplir con un listado mínimo de características que tiene que tener nuestra página web. Especificamos aquí de forma resumida en qué parte de la web se encuentra cada una de estas características, en forma de lista de comprobación o *check-list* para facilitar la evaluación de la asignatura:

- **Un menú desplegable:** se encuentra en la navBar (figura 2.3).
- **Galería de imágenes:** en la página interna de la sección front-end implementaremos *swiper.js* a tal efecto (figura 2.4).
- **Galería de vídeo:** se encontrará en el apartado “Quienes somos”, justo debajo de la navBar (ver anexo 6.7).
- **Un formulario:** Se encontrará en la página de contacto (figura 1.7).
- **Una ubicación:** Se encontrará en la página interna “quienes somos” muy cerca de la footer (figura 6.7).
- **Uso de un número mínimo de librerías:** En el tercer requisito que viene a continuación lo explicaremos detalladamente: tenemos proyectado el uso de *swiper.com*, *animate.style* y de *chartjs*.
- **Uso de una tabla:** La empleamos en la página interna de front-end (figura 6.3) para comparar características entre los tres frameworks de front-end.
- **Uso de un gráfico de librería:** Como comentaremos en el tercer requisito vamos a usar *chartjs* para la sección de IA y big data.

El **tercer requisito** que tenemos es que para codificar la web hay que escoger entre una o varias de estas librerías (que si bien pueden usar JavaScript, en el proyecto no se nos permite hacer uso directo del mismo de acuerdo con el primer requisito):

- [animate.style](#): una biblioteca de CSS que incorpora animaciones visuales. Tiene keyframes que se pueden añadir fácilmente con una línea de css, con la propiedad “animation”. Se puede configurar su duración mediante la propiedad “animation-duration”.
- [wowjs.uk](#): Permite usar animaciones de Animate, que serán invocadas al hacer scroll. Solo las invoca si son visibles, con lo cual se gana en rendimiento porque consume menos recursos del navegador.
- [popper.js](#): herramientas para crear menús desplegables (y efectos interactivos) bonitos.
- [swiperjs.com](#): para crear sliders y galerías de imágenes interactivas.
- [chartjs](#): se usa para crear gráficos interactivos.

Hemos proyectado el uso **de tres** de estas librerías en nuestra página web (que con el desarrollo de las páginas internas se podrán ampliar después de la primera evaluación, cuando las vayamos a codificar) y clarificamos su uso a continuación:

- [animate.style](#): Hemos usado la librería de clases CSS *animate.style* para configurar la animación de salida de los elementos del menú desplegable de la navBar, disponible en todas las páginas de la web.

- **swiperjs.com**: La vamos a usar para crear una *galería de imágenes* que nos permita cambiar de forma agradable entre las distintas imágenes de las tres tecnologías front-end (react, angular y vue) en la sección front-end de la página interna mostrada en anexo ([link](#)), también observable en la figura 2.4:
- **chartjs**: Crearemos un *gráfico interactivo* en la página interna de ia y big-data (sección 6.5) que se va a mostrar al hacer hover en el ícono de Seaborn o de Plotly, **sustituyendo** ese ícono momentáneamente. Es decir, en el mock-up adjunto en anexo no se reflejará este gráfico, porque simplemente debería ocupar el mismo espacio que el ícono al que sustituye y ello nos obligaría a hacer una exportación del lienzo por partida doble.

¿Qué diferencias fundamentales hay entre los tres frameworks?

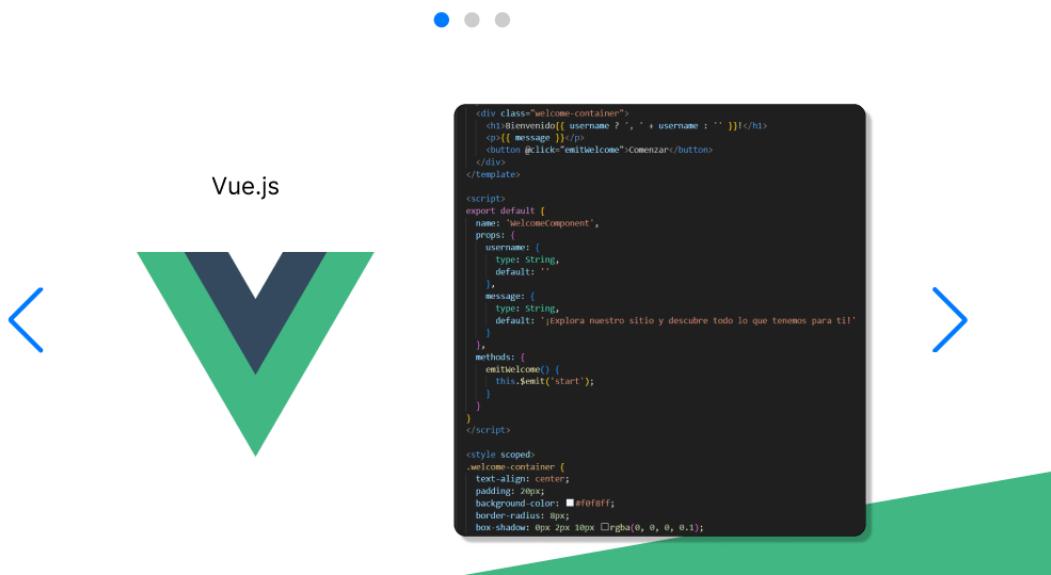


Figura 2.4: Uso de swiper en la página interna de tecnologías front-end de la web. Flechas y paginadores se añadirán automáticamente por la librería.

El **cuarto requisito** es que la página web **no tiene que ser responsive**. Por lo tanto, el uso de media queries no se puede valorar. Por ende, esta web se ha optimizado para su visionado en pantallas de ordenador de resolución 1920 x 1080 formato (fullHD) con el zoom del navegador al 100 %. En esos casos la web va a ser prácticamente igual -o extremadamente parecida- a su maquetación previa en Figma¹.

¹¡Cuidado! Al no usar media queries la página no se va a ver bien en móviles ni en tablets. En estos casos de uso, que no precisan adecuación en este caso, aparece una línea vertical blanca en la parte derecha de la pantalla. Asimismo, es probable que con dispositivos de mayor resolución horizontal a 1920 píxeles de ancho, la página se vea mal.

El **quinto requisito** no viene de fuera, sino que nos lo hemos marcado nosotros. Hacer una página web sencilla, usable y atractiva para el usuario. Minimalista, con combinaciones de colores atractivos y efectos visuales con hover y las librerías a las que se nos acota su uso en el segundo requisito.

El **sexto requisito**, aunque implícito, consideramos que es la accesibilidad de la página web. Dado que esta memoria forma parte de la asignatura de diseño de interfaces, cuyo objetivo es tener la experiencia de usuario en el foco, creemos que la web debe ser no solamente atractiva visualmente sino también fácilmente navegable. Los elementos deben estar en localizaciones típicas para facilitar la presentación de la empresa y el contacto de los clientes. Es por eso que hemos optado por una barra de navegación con cuatro elementos, con un menú desplegable que la deja limpia y organizada en todo momento. También en el **footer** de la landing page, o de cualquier página interna, permitimos acceso a las distintas páginas internas de la web.

2.1.3. Lenguaje en la parte cliente

En la parte de cliente la web se desarrollará en HTML5 y en CSS3. Solamente siendo asistidos por las librerías mencionadas en el segundo requisito del apartado anterior, que nos permitirán mejorar la experiencia de usuario.

Se han usado las etiquetas `<section>` y `<article>` de HTML5 para mostrar semánticamente las distintas partes en la landing-page (que al momento de hacer esta memoria es la única parte de la web que debe estar codificada), pero también se han combinado con el uso de las etiquetas antiguas de anteriores versiones de HTML (`<div>`), dado que no siempre las distintas divisiones de los bloques de diseño que se han creado responden a regiones con significados semánticos.

Por ejemplo, esto se hace evidente en el `index.html` cuando programamos el contenedor que contiene los gradientes que vimos en la figura de la sección de front-end de nuestro index, inspirado en Apple [ver figura](#). Para programar este contenedor, en realidad, tenemos un total de 3 `div`, ninguno de los cuales es semántico: el primer `div` sirve para tapar un vestigio del contenedor de la onda de la landing-page, el segundo para incluir el primer gradiente de blanco a azul y el tercero para incluir el segundo gradiente de azul a negro.

Por el contrario, el **cuarto div** es el que es distinto a los anteriores, pues va a albergar en su interior, de forma anidada, todo el contenido de la sección de tecnologías front-end y se va a superponer con los `div` anteriores mediante posicionamiento absoluto. Es este último `div` el que incluye las cartas de las tres tecnologías que sí van a requerir etiquetas semánticas, tanto las etiquetas `<section>` como `<article>`. En la proxima imagen mostramos la correspondencia entre código html e interfaz, se-

parando sus componentes semánticos y no semánticos según acabamos de describir:

Figura 2.5: Detalle del código index.html en donde se muestra la coexistencia necesaria entre etiquetas html y html5 antiguas para diferenciar el layout de la semántica.

————— NS: Contenido No Semántico | S: Contenido Semántico ————



```
<!-- SECCIÓN 2 (S2): tecnologías frontend -->
<section id="tec-front-end">
    <div></div> <!-- div:first child -- contenedor INVISIBLE que tapa la linea vestigial que aparece debajo del contenedor onda-->
    <div></div> <!-- div:second child -- contenedor gradiente de blanco a azul-->
    <div></div> <!-- div:third child -- contenedor gradiente de azul a negro-->
    <div>      <!-- div:last child -- contenedor gradiente que tendrá texto y las 3 cards-->
```

NS

```
<!--Contenedor con información Frontend (section:FirstChild)-->
<section>
    <h1 class = "tituloSeccion">Front-end</h1>
    <p class = "subtituloSeccion">Te llevamos las últimas tecnologías basadas en componentes para tu negocio. Refactorizamos tu código en HTML, CSS y JavaScript o hacemos el cliente de tu web desde cero. Consigue soluciones escalables, mantenibles y modulares.</p>
</section>

<!--Contenedor con las tres cards (section:LastChild) -->
<section>
    <article class="cartaFrontEnd">
        <a href="#"></a> <!--redirigirá a Frontend (pagina interna)-->
        <p>Librería de JavaScript enfocada en la interfaz de usuario: flexible, minimalista y con un mayor control sobre la arquitectura.</p>
    </article>
    <article class="cartaFrontEnd">
        <a href="#"></a> <!--redirigirá a Frontend (pagina interna)-->
        <p>Framework completo de desarrollo web, basado en TypeScript, estructurado y con muchas herramientas integradas, ideal para aplicaciones empresariales.</p>
    </article>
    <article class="cartaFrontEnd">
        <a href="#"></a> <!--redirigirá a Frontend (pagina interna)-->
```

S

2.2. Planificación y fases del desarrollo

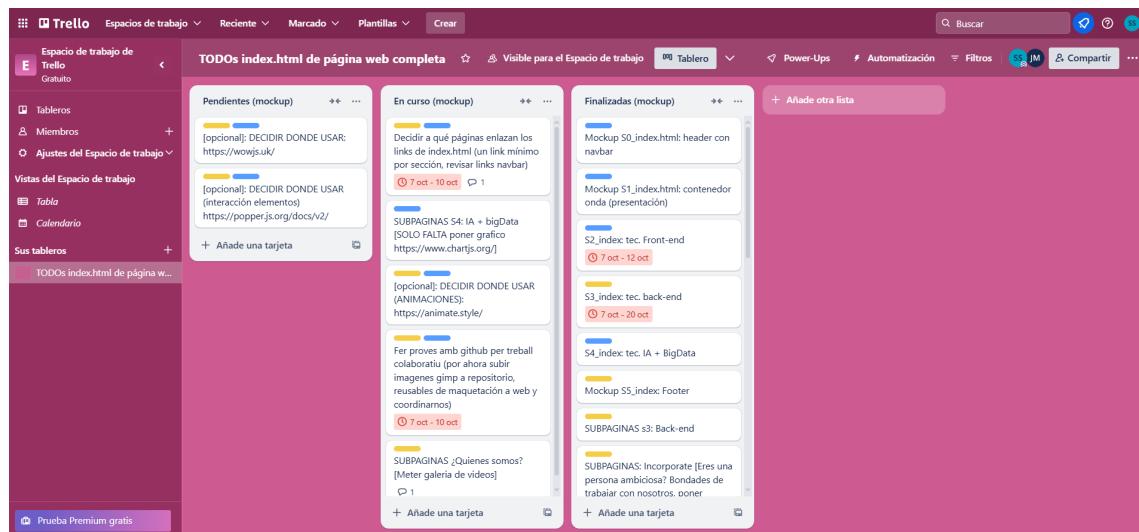
2.2.1. Fase 1*: Uso de Trello para distribuir tareas

Hemos utilizado Trello para distribuirnos las tareas del mock-up. Lo hemos usado de tal modo que existe una etiqueta o color (amarilla -de Jorge- o azul -de Santi-) para cada persona. Luego cada tarea queda clasificada en tres bloques: pendientes, en curso y finalizadas.

Las tareas de desarrollo del *mockup* se asignaron a cada uno de nosotros en una videollamada (existen tareas individuales y tareas conjuntas) de la que salió la figura 2.6. Nótese que en esta captura del dashboard de Trello no existe una repartición de tareas de la codificación de la web, solamente del *mockup*. Esto no es casual, porque cada persona ha sido responsable de codificar cada sección que se le asignó en el *mockup*. Es por ello que no ha sido necesario repetir otro dashboard más para ello.

No se han puesto fechas en muchas de las tareas a repartir, dado que en esencia hemos funcionado con sprints repartidos a lo largo del curso.

Figura 2.6: Detalle del dashboard de nuestro Trello para la distribución de tareas.

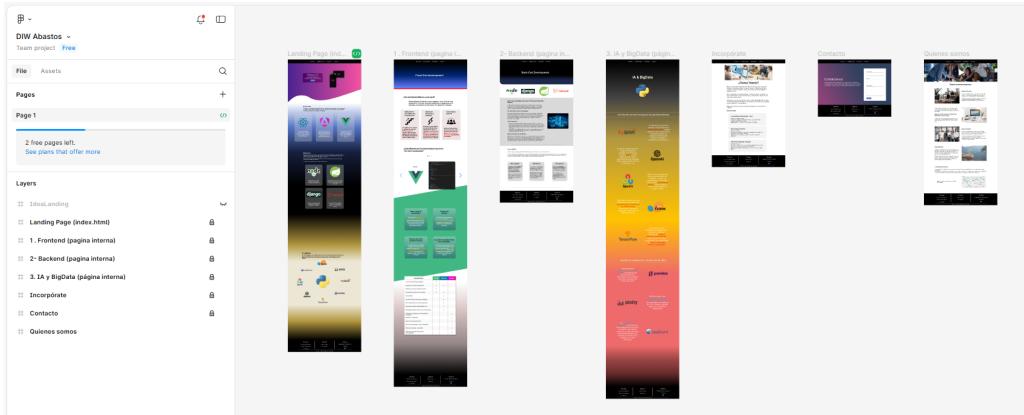


NOTA: El link de este dashboard de Trello está [aquí](#) (privado).

2.2.2. Fase 2*: mockup con figma

Para hacer la maquetación de Figma hemos usado lienzos de 1920 píxeles de ancho por todos los píxeles que sean necesarios en vertical. Esta es una captura de la vista general de figma:

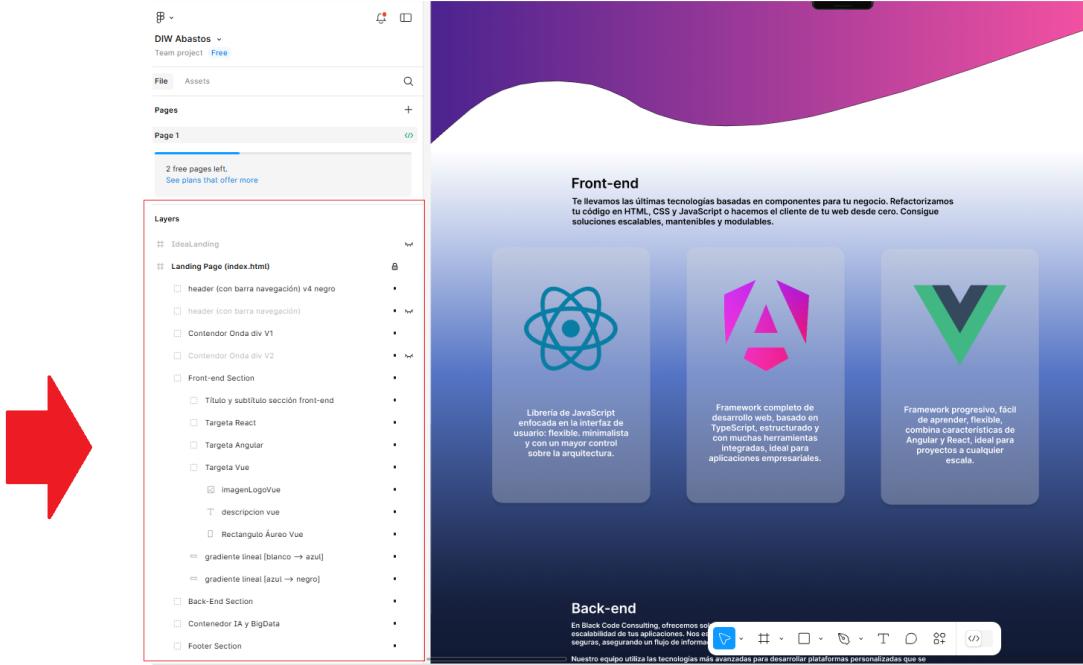
Figura 2.7: Detalle del dashboard de nuestro Figma, con el que hemos hecho el mockup de la web completa.



Figma se ha probado como una herramienta sorprendentemente eficiente y útil para el trabajo colaborativo. Permitiendo trabajar incluso a la vez viendo como tu compañero va trabajando en su parte.

Hemos trabajado de la forma más organizada posible. Cada uno de los elementos que componen los lienzos del mockup se han ido anidando de forma parecida a como lo haríamos con las distintas etiquetas html dentro de distintos bloques de una página html. Figma nos ha permitido el prototipado rápido de varias secciones, habilitándonos a hacer pruebas de superposición de varias versiones, pudiendo volver a una antigua sin estar obligado a tener varios lienzos (ver detalle en rojo, como se estructura por ejemplo la landing page en la figura siguiente):

Figura 2.8: Detalle del trabajo por capas y bloques que hemos hecho en Figma para la landing page.



En la figura 2.8 anterior se puede ver, por ejemplo, que en este lienzo de Figma tenemos dos “contenedor Onda div” distintos, uno superpuesto al otro; pero solo el que consideramos definitivo se muestra. También podemos ver como la sección de tecnologías front-end de la landing page se ha organizado de forma jerárquica, anidada. Cada tarjeta tiene su bloque o carpeta en donde se organizan sus elementos, por ejemplo.

Finalmente mencionar que Figma tiene una forma extremadamente intuitiva de definir las prioridades en los solapamientos (el equivalente al z-index en el CSS de una web): si tenemos dos elementos que ocupan el mismo espacio en el lienzo, solo es necesario poner antes en la lista aquel elemento que queremos que salga por encima. En definitiva, solo se trata de ordenar la lista para definir el orden en la superposición.

NOTA: Aquí está [el enlace de nuestro mockup](#) en Figma (se requiere contraseña). Alternativamente el mockup se puede consultar en el anexo (capítulo 6 de la memoria).

2.2.3. Fase 3*: GitHub y desarrollo colaborativo

Para compartir código y desarrollar las distintas partes de la página web hemos utilizado GitHub. Por ejemplo, para codificar el index.html, hemos trabajado por ramas de característica o “feature branches”. Cada uno ha desarrollado una parte de su archivo html y parte de su CSS, en remoto, utilizando esta técnica de ramas de “característica”(no se nos ocurre una adaptación mejor que su traducción literal del inglés): es decir, ramas que aglutan los cambios en los archivos para generar una característica en concreto que luego, al estar bien desarrollada, integraremos en la rama main o de producción. Por ejemplo, si uno quiere desarrollar la sección de tecnologías front-end y no tiene todavía el repositorio en local hará:

```
# clonar el repositorio en local
git clone https://github.com/blackcub3s/proyectoDesarrolloInterfaces.git

# trabajar previamente con el index.html y el css
# luego anyadirlos al area de preparacion
git add -A

# creamos rama para aglutinar los cambios
git branch seccioFrontEnd

# cambiamos a la rama que acabamos de crear
git checkout seccioFrontEnd

# guardamos los cambios como nodos dentro de
# la rama con la que desarrollamos.
git commit -m "commit 1"
git commit -m "commit 2"
# [...]
git commit -m "commit n"

#subimos los cambios a la rama remota
# (si no existe se crea automaticamente).
git push origin seccioFrontEnd
```

Como vemos en el ejemplo anterior, al estar contentos con la rama ”seccioFrontEnd” se ha subido esa rama a una rama remota a GitHub, con el mismo nombre. Lo que entonces nos queda por hacer, después de este código, es hacer una **pull request** para solicitar integrar esos cambios de la rama-característica remota (origin seccioFrontEnd) a la rama principal remota (origin main), que será ya la última iteración del código de producción. Con la finalidad de que el compañero de equipo pueda ver los cambios que se quieren integrar o que se han integrado, de una sola

vez, al hacer la pull request desde la interfaz de GitHub hay una opción que nos permite hacer un “squash and merge”, que es simplemente un merge de la rama feature con la rama main pero que compacta convenientemente todos los commits de la rama feature en un solo commit.

El motivo por el cual no integramos directamente el código en la rama main es porque hemos definido reglas (rulesets) en el repositorio para impedirnos subir la rama main local directamente a la rama main remota, por seguridad, para que nadie destruya el código sin querer.

Una vez la rama característica “seccioFrontEnd” ha sido integrada en la rama main o principal y se ha borrado la rama remota de la característica ya integrada, ya tenemos una nueva hornada del código con otra sección funcionando correctamente y guardado en el repositorio remoto.

Luego, para seguir desarrollando en local, cualquiera de nosotros dos podía hacer `git pull origin main` desde su ordenador para descargarse la rama main en local, y hacer una ramificación con la nueva característica que desease añadir.

Por terminar con el ejemplo anterior de la ramaFrontEnd, eso es lo que hemos hecho para llegar a la figura 2.9. Se puede ver que se ha usado esta técnica para integrar todos los cambios de la sección de tecnologías de front-end del index.html en la rama main:

Figura 2.9: Vista general de la unión de la rama característica en la que se desarrolló la sección de tecnologías front-end del índice.html y como esta se integró en la rama main a través de una pull request.

The screenshot shows a GitHub repository page for 'blackcub3s / proyectoDesarrolloInterfaces'. The 'Pull requests' tab is selected. A modal window is open, prompting the user to 'Label issues and pull requests for new contributors'. It includes a message: 'Now, GitHub will help potential first-time contributors discover issues labeled with good first issue' and a 'Dismiss' button. Below the modal, there's a 'Filters' dropdown set to 'blackcub3s/proyectoDesarrollo... on Nov 7'. A red arrow points to the commit history of a pull request. The commit history shows three commits from the 'seccio front end' branch merged into the 'main' branch. The commits are: 'main --> seccioFrontEnd' (by 'Arr... #2 by'), 'You commented on and opened this pull request' (by 'blackcub3s'), and '#1 by blackcub3s was merged yesterday' (by 'blackcub3s'). The right side of the screen shows standard GitHub search and filter controls: Labels (9), Milestones (0), New pull request, Author, Label, Projects, Milestones, Reviews, Assignee, Sort, and a ProTip! notification.

Al clicar en el link marcado con la flecha en rojo, en la anterior captura, podemos ver que la rama de característica “SeccioFrontEnd” tenía tres commits y que luego

quedó integrada en la rama main:

Seccio front end #1

Figura 2.10: Vista ampliada de la unión de la rama de característica SeccioFrontEnd del índice y como sus tres commits se integraron en la rama main

Se pueden ver los cambios hechos en los archivos con esta integración en concreto si se accede a este [link](#). Alternativamente se puede observar uno de los archivos que han cambiado, por ejemplo, el index.html:

Figura 2.11: Muestra parcial de la integración de cambios en el pull request de tipo “merge squash” de la rama de la sección de tecnologías front-end hacia la rama main, para el archivo index.html

Además, GitHub nos ha proporcionado la posibilidad de compartir código con mucha rapidez. Por ejemplo, en la figura 2.12 vemos un fragmento en que uno de

nosotros (Jorge) desarrolló una característica de la web, subió ello a la rama remota Footer, y el otro integrante (Santiago -blackcub3s-) hizo algunos cambios antes de luego ya decidir hacer la pull request definitiva a la rama main o rama de producción. Análogamente en la figura 2.13 podemos ver todas las pull requests que hemos ido haciendo en el proyecto, al menos hasta el día 13 de noviembre de 2024 que es cuando este párrafo fue escrito.

Figura 2.12: Trabajo colaborativo entre los integrantes del grupo

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base: main ▾ compare: Footer ✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

-o 3 commits

4 files changed

2 contributors

Commits on Nov 10, 2024

- Sección footer programada, HTML + CSS. Versión sin barras verticales...** f485528
- Footer acabado, Versión con HTML+CSS, a priori ningun cambio por hacer** 6a0f576
- canviat h3 per h1 per consistencia semàntica. Afegit maxwidth a la fo...** 715e925

Figura 2.13: Pull requests a origin main (rama remota de producción) hechas durante el proyecto de desarrollo -hasta 13nov24-.

blackcub3s / proyectoDesarrolloInterfaces

Code Issues Pull requests Actions Projects Security Insights Settings

Type (I) to search

Filters Q ispr isclosed

Clear current search query, filters, and sorts

0 Open 8 Closed

Author Label Projects Milestones Reviews Assignee Sort

- eliminado estilo tituloSeccion, cuyo margin-bottom se sumaba sin qu... #6 by blackcub3s was merged yesterday • Review required
- Backend Section #7 by Jorgecm723 was merged yesterday • Approved
- la big data #6 by blackcub3s was merged yesterday • Review required
- Anadidos los archivos necesarios para programación apartado backend #5 by Jorgecm723 was merged 3 days ago • Approved
- Footer #4 by Jorgecm723 was merged 3 days ago • Approved
- creat desplegable navbar i afegides animacions 'fadeInDown' animate #3 by blackcub3s was merged 4 days ago • Review required
- Arreglo marges dispositius petits ihovers #2 by blackcub3s was merged 5 days ago • Review required
- Seccio front end #1 by blackcub3s was merged last week • Review required

New pull request

Finalmente hay que mencionar que todos los selectores CSS descendientes deberán empezar por el identificador de cada sección de la landing page o, en su defecto, usar etiquetas que no se usan en ninguna parte más de la página (footer, nav, etc.), siendo esto ampliable al resto de las páginas internas de la web que entregaremos más adelante. Así se evitarán choques de estilos al desarrollar la misma web varias personas.

Para evitar conflictos en el trabajo colaborativo se tomó la decisión de usar tantos archivos CSS como secciones (así habría menos archivos en los que dos personas podríamos estar trabajando simultáneamente, disminuyendo los conflictos al juntar ramas de “característica” con la rama main de producción). Tomar esa decisión ha sido crucial para evitar conflictos, ya que la suma de líneas de código que hay en los CSS de la landing page llega casi a 1000. Con un archivo tan largo podríamos haber estado ambos trabajando en la misma parte del documento produciendo problemas de integración al hacer merge. En cambio, cada uno trabajando en secciones separadas, con un archivo CSS distinto asociado a cada sección este problema se ha evitado. Si se desea ver la versión por archivos separados se puede ver la rama de GitHub *cssSeparados*.

Evidentemente, a recomendación de Sonsoles, se ha tomado la decisión de juntar al final del desarrollo en la rama de producción (rama main) todos estos archivos CSS en uno solo (*estilos.css*) por simplicidad y buenas prácticas en el desarrollo. **Es esta rama, la rama main, la que entregamos a términos de evaluación.**

Aquí los links públicos al proyecto:

Link al repositorio	https://github.com/blackcub3s/proyectoDesarrolloInterfaces
Página desplegada	https://blackcub3s.github.io/proyectoDesarrolloInterfaces

Cuadro 2.1: Enlaces importantes del proyecto.

Desarrollo

3.1. Diseño

El **proceso de diseño** fue tal que el layout se organizó de forma que se tuviese en mente muchísimo la *jerarquía visual* y la *disposición* de los elementos y la *iconografía*. En la landing page, por ejemplo, tenemos iconos muy grandes y dispuestos simétricamente, que llaman la atención e invitan al usuario a clicar y a redirigirse por las páginas internas. Mediante los gradientes de color conseguimos que hacer scroll sea una delicia y que al usuario le invite a moverse de arriba a abajo, revelando nuevos elementos que de otra forma permanecerían ocultos. Eso ayuda a que el usuario vea más parte de la página de lo que en condiciones normales fuese a ver.

En relación a la **interfaz de usuario (UI/UX)** la elección de *tipografía* se hizo sin complicaciones: usamos roboto, que es la que utiliza Figma por defecto al ser considerada óptima para el diseño. Luego, al codificar, cargamos la fuente correspondiente de google fonts y la definimos en el CSS.

La *elección de colores* no se hizo tomando en cuenta ningún criterio profesionalmente técnico. Simplemente escogimos aquello que veíamos que funciona en otras páginas web: como ya se ha comentado en el apartado comparativa 1.3) tomamos un razonamiento inductivo y seleccionamos aquello que nos parecía visualmente atractivo al comparar múltiples webs del ámbito. Sí querríamos comentar sobre la elección de colores de una de las secciones iniciales y más vistosas de la landing page: el azul y el doble gradiente con distintos tonos de este color.

El color azul y los gradientes, a parte de transmitir confianza (como veremos en el siguiente párrafo), simbolizan el apartado visual de una web, su front-end. Por el contrario, el color negro de la sección de back-end simboliza la parte “invisible” del servidor: una caja negra inaccesible y opaca para los usuarios. Finalmente, el color escogido para la sección de ia y big data (amarillo y blanco amarillento) emula los colores del sol al amanecer, por referirse a un conjunto de tecnologías emergentes y potentes (además que justamente el propio ícono de Python actúa como si fuese la fuente de iluminación de los iconos que lo rodean, como veremos luego en la figura 3.3).

Si el lector quiere profundizar en el uso del color y los porqués asociados a

este uso, puede ver una revisión sistemática¹ publicada en 2015 por un autor de la universidad de Rochester [1] en el que se resumen las correlaciones existentes entre el visionado de determinados colores y el funcionamiento cognitivo. Le prometemos que no va a quedar indiferente. Ante todo, el mismo autor destaca la realidad de su campo: las conclusiones a las afirmaciones hay que hacerlas con cautela, dado que la psicología del color no es una ciencia exacta; pero también muestra en la tabla 1 de su estudio (“*table 1: Research on color and psychological functioning*”) varias conclusiones interesantes de la evidencia previa. Una de ellas, extraída a partir de múltiples estudios científicos comprendidos entre 2007 y 2014, dice que **al mostrar páginas en azul y páginas en verde a los usuarios de páginas web, éstos tenían tendencia a confiar más en las páginas que estaban definidas en color azul**. Por lo tanto, con el azul aparentemente se puede conseguir un efecto psicológico de confianza; sea en mayor o menor medida, eso es positivo para una web que quiere mostrar un negocio de consultoría

Figura 3.1: Revisión de estudios científicos adaptada de [1]

Area of research	Central finding	Example	References
Color and Store/Company Evaluation	Blue stores/logos have been shown to increase quality and trustworthiness appraisals	Participants rated websites featuring blue (relative to green) as more trustworthy	Yüksel, 2009 ; Lee and Rao, 2010 ; Alberts and van der Geest, 2011 ; Labrecque and Milne, 2012 ; Ridgway and Myers, 2014 (cf. Barli et al., 2006 ; Chebat and Morrin, 2007)

Otro aspecto importante a tratar es la **Interactividad** de la página. Hemos utilizado animaciones para el menú desplegable servicios. Para ello hemos usado la librería CSS animate.style. Estas animaciones están solo en el index.html, no en el mockup, que es completamente estático.

Para *cada una de las “cards” del front-end* hemos usado la pseudoclase :hover para hacer que al pasar por encima con el ratón aparezca un sombreado más marcado que el que ya tienen en condiciones normales y, además, hemos conseguido que las imágenes de cada una de las cards (los logos de vue, angular y react) se hagan más grandes al pasar por encima. Análogamente, *en cada una de las cards del back-end* la pseudoclase :hover se ha utilizado para definir un color más claro en la parte

¹artículo que aglutina y resume múltiples estudios científicos revisados por pares

interior de sus bordes al pasar por encima con el ratón y también aumentar los tamaños de los iconos.

Además, *todos y cada uno de los iconos de la sección de IA y big data* -los que están dispuestos en un círculo y el que está en la posición central- tienen una animación de la propiedad filter para conseguir un sombreado más marcado al pasar por encima con el ratón. A continuación podéis ver el efecto cuando el ratón no pasa por encima y cuando la pseudoclase :hover se activa al pasarlo (más adelante, en la parte de codificación se puede ver como se ha conseguido esto, a nivel técnico, en la figura 3.8):

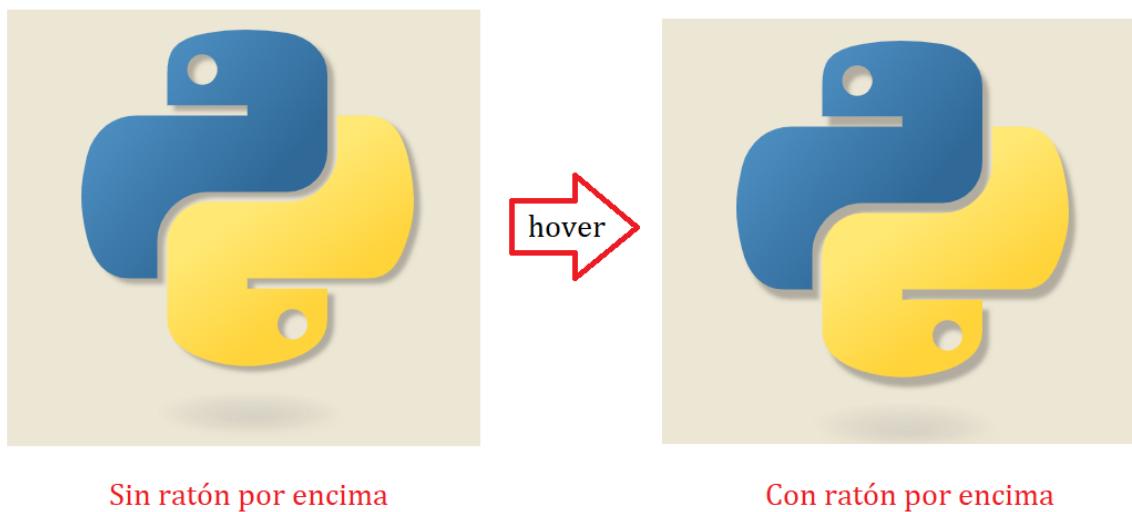


Figura 3.2: Hover encima del ícono de Python de la landing page, en la sección de tecnologías IA y BigData, definido por el id de CSS #pythonLanding e id #pythonLanding:hover (solución técnica en figura 3.8).

Es menester mencionar que se ha tomado mucho esmero en conseguir los sombreados de la sección de IA y BigData de la landing page. Como se puede ver en la figura 3.3 hemos conseguido emular exactamente la maquetación con Figma. No solamente las sombras de los iconos de las librerías de python dispuestos en círculo (que toman una dirección radial en relación al ícono central Python, que es como un sol) sino también las sombras interactivas que se alargan cuando el usuario pasa el ratón por encima, de nuevo, mediante el uso de la pseudoclase :hover:

Figura 3.3: Muestra de la sección ia-big-data capturada directamente del index.html. ([Consultar la web](#)) para ver el sombreado dinámico que se ha animado mediante la pseudoclase hover: hemos variado la propiedad `filter`, pasándole la función `drop-shadow` y mediante el uso de la pseudoclase `:hover`. El posicionamiento de cada ícono y tamaño es exactamente calcado a Figma. Se ha hecho un trabajo muy minucioso de réplica con el mockup. El ícono de Python actúa como foco de iluminación.



3.2. Funcionalidad

Para esta web **no se han hecho pruebas de usabilidad** con usuarios, dado que sirve casos de uso muy restringidos; y solo tiene codificado su `index.html`.

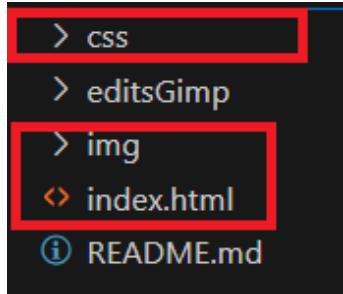
Las interacciones de los usuarios con la interfaz u otras características clave ya se describen en las demás secciones, así que no contestaremos a más aspectos en este apartado para evitar ser redundantes.

3.3. Codificación

Antes que nada hablar de la estructura del proyecto. Vamos a tener el `index.html` y luego una subcarpeta para las imágenes (`img`) y otra para el `css` (`css`), tal y como vemos en la figura 3.4. Al ampliar con más páginas `html` las tendremos todas en la

raíz del proyecto, al mismo nivel que ahora tenemos el index.html.

Figura 3.4: Estructura de la carpeta del proyecto (en rojo).



En la figura anterior vemos que hay otra carpeta *editsGimp*² y *README.md*³. Estos ficheros no tienen impacto en la interpretación de la página web en el navegador (ver notas al pie).

Por ahora, dado que para la primera evaluación solo era necesario codificar el *index.html*, para la sección de codificación de la memoria hablaremos íntegramente de este archivo y de su CSS asociado.

Para empezar, hablaremos del body. Dentro del body de nuestra página *index.html* tenemos:

- A: al principio el `<header>` con `<nav>`, que se replicará luego en todas las demás páginas internas.
- B: Al final el `<footer>` que también se replicará en las demás páginas internas, con información de contacto y acceso a las distintas secciones de la página web (es redundante con la navbar, para asegurar la accesibilidad).
- C: En medio tenemos las `<section>`, de las que hay cuatro. Cada una de ellas corresponde con una parte visible de la página, situada entre la `navbar` y la `footer`. Como vemos en la figura 3.5 para cada una de esas secciones se ha utilizado una etiqueta semántica `<section>` (una de las nuevas etiquetas definidas en HTML5), cada una de ellas con un id asociado -¡dado que no son únicas!-. Dentro de cada section usamos distintas etiquetas tales como `h1`, `p`, `article` o `div` (dependiendo del significado semántico o de organización que se le quiera dar⁴). Estas etiquetas, especialmente `section` y `article`, deberían ayudar a optimizar el SEO ya que son nuevas etiquetas de HTML5.

²Ediciones de iconos que hemos hecho con Gimp para mostrar como hemos eliminado los canales alpha para habilitar transparencias.

³Fichero para mostrar información del repositorio de GitHub con el que trabajamos en remoto en equipo.

⁴section y article, nuevas etiquetas de HTML5 dotan a la etiqueta de significado semántico; div, etiqueta antigua heredada de versiones anteriores de html, no da ese matiz (de ahí que en el proyecto decidimos que coexistan, según convenga.)

Para asegurar que no haya errores en el código del *index.html* hemos usado el prevalidador de html de w3schools.

Figura 3.5: Secciones minimizadas en las que estructuramos el body del *index.html*, nuestra landing-page. Cada sección tiene un id que hemos usado para dar estilos mediante selectores descendientes, a excepción de **header** y **footer** que en el mismo *index.html* no se repiten y no requieren id.

```
<body>
  <!-- SECCIÓN 0 (S0) -- HEADER: barra de navegación-->
  <header>
    <nav> ...
    </nav>
  </header>
  <!-- SECCIÓN 1 (S1): contenedor-onda: texto de presentación e imágenes de dispositivos electrónicos -->
  <section id="contenedor-onda"> ...
  </section>
  <!-- SECCIÓN 2 (S2): tecnologías frontend -->
  <section id="tec-front-end"> ...
  </section>

  <!-- SECCIÓN 3 (S3): tecnologías de backend -->
  <section id="tec-back-end"> ...
  </section>

  <!-- SECCIÓN 4 (S4): tecnologías de IA & BigData -->
  <section id="IA-bigData"> ...
  </section>

  <!-- SECCIÓN 5 (S5): footer -->
  <footer>...
  </footer>
</body>
```

Al codificar el CSS se ha intentado ser lo máximo ordenado posible, sin abusar de las clases ni de los *ids*. Se ha usado muchísimos selectores descendentes para evitarlo y el uso de las pseudoclases `:first-child()`, `:nth-child()` y `:last-child()` ha sido recurrente y necesario para conseguirlo. Con solo los ids de cada una de las 5 secciones y los selectores descendentes y descendientes directos hemos podido codificar el CSS de prácticamente todo (especialmente secciones ia, front-end y navbar).

Fuera del body, antes de este, tenemos definida la etiqueta **head** de la página: ahí es donde hemos cargado todos los estilos customizados que hemos definido en archivos aparte fuera del HTML, además de la tipografía de google fonts y del link a la CDN de *animate.style* (ver figura 3.6).

```

<head>
    ...
    <!--Cargamos la fuente roboto de google fonts -->
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

    <!--Cargamos el CSS de animate.style para hacer animaciones, p. ej menu desplegable de la navbar -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>

    <link rel="stylesheet" href="css/estilosglobales.css">
    <link rel="stylesheet" href="css/barraNavegacion.css">
    <link rel="stylesheet" href="css/contenedorOnda.css">
    <link rel="stylesheet" href="css/tecnologiasFrontEnd.css">
    <link rel="stylesheet" href="css/tecnologiasBackEnd.css">
    <link rel="stylesheet" href="css/tecnologiasIABigData.css">
    <link rel="stylesheet" href="css/footer.css">
</head>

```

Se juntará en un solo archivo en producción

Figura 3.6: El head de nuestra página: definimos la tipografía, tomamos el link a los estilos de animate.style y cargamos nuestros estilos customizados -que en producción, en la rama main, están juntos en un *sol archivo estilos.css* por mantenibilidad-.

En nuestros archivos CSS customizados hemos definido, en estilosGlobales, dentro de la pseudoclase :root todas las variables CSS que hemos previsto tener que reutilizar (ver figura 3.7)⁵:

⁵La codificación del CSS ha sido en “vanilla” CSS: codificación directa. No se han usado ni preprocesadores como SASS, ni frameworks como Bootstrap ni frameworks por componentes como React, Vue o Angular. Solamente la librería animate.style para la animación del menú desplegable de la *navBar*.

```

:root {
    /* ---- paleta de colores ---- */
    --colorBarraNavegacion: □rgb(0, 0, 0);
    --colorBarraNavegacionHover: □rgb(170, 170, 170);

    --colorAzulBoton: □#5474C4;           /*contenedor onda*/
    --colorAzulBotonHover: □#3652a3;       /*contenedor onda*/
    --colorVerdeVue: □#41B883;           /*contenedor onda*/
    --colorGris: □#A3A3A3A3;             /*contenedor onda*/
    --colorGrisTransparent: □rgba(38, 38, 38, 0.449); /*contenedor onda*/
    --colorFinGradiente: □#F652A2;       /*contenedor onda*/
    --colorInicioGradiente: □#4b238f;     /*contenedor onda*/

    --grisTransparenteCard: □#f4f4f43e;   /*parte front end landing*/
    --blancoGradiente: □#FFFFFF;          /*parte front end landing*/
    --azulClaroGradiente: □#5474C4;       /*parte front end landing*/
    --azulOscuroGradiente: □#16203E;      /*parte front end landing y back-end landing*/
    --negroGradiente: □#000000;           /*parte back-end landing e iaBigData*/

    --doradoGradiente: □#B39840;          /*parte IA bigData*/
    --blancoAmarillo: □#ECE6D4;           /*parte IA bigData*/
    --negroFooter: □black;                /*footer*/

    /*otras variables susceptibles
    de cambio en función de decisiones futuras*/
    --MaximaAnchuraBarraNavegacion: 600px;
    --MaximaAnchuraContenedorOnda: 1000px;
    --MaximaAnchuraTextosEncabezados: 1000px;
    --MaximaAnchuraCards: 1400px;
    --MaximaAnchuraFooter: 1200px;
    --tiempoTransicionEstandar: 0.1s;
    --aumentoTamanoEstandar: 1.02; /*aumentos para las imágenes al hacer el hover (aumento del 2 por ciento)*/

    /*variables para centrar en centro las imágenes satelite de python en contenedor ia*/
    --X: 23em;
    --Y: 27em;
}

```

Figura 3.7: Definición de variables css dentro de la pseudoclase `:root` para colores de la web, anchos de secciones usados con la propiedad *max-width* y, finalmente, las variables `-X` e `-Y`, que son el centro del círculo de la sección IA

Como también comentábamos en el apartado diseño de esta memoria, los **iconos de la sección de IA y bigData** -los que están dispuestos en un círculo, volver a ver figura 3.3- tienen una animación de la propiedad `filter`⁶ que se activa cuando pasamos el ratón por encima. Ahora *comentaremos los aspectos técnicos de código relacionados con ella*:

En primer lugar, mencionar que al animar esa propiedad usamos también la propiedad `transition` con el valor `filter` y también *ease-in-out* para hacer que la transición producida por el `hover` en el cambio de valor de filter sea suave tanto al poner el ratón encima como al quitarlo. Esta animación con la pseudoclase `:hover` permite que el usuario sienta que el ratón produce un cambio de inclinación de la iluminación en el ícono (podéis regresar a la figura 3.2 de la sección diseño para ver como esto se aplica al ícono de Python). No ha sido posible manipular la altura

⁶La propiedad `filter`, al pasarle la función “*drop-shadow*” con sus argumentos es como la propiedad `box-shadow`, pero que a diferencia de ésta usa el contorno de la imagen vectorial para generar la sombra en lugar del rectángulo html que la alberga.

en el eje de las Z del icono de Python (que es como un sol que ilumina los iconos satélite) porque el posicionamiento de este icono -y de todos los demás iconos en esta sección, de hecho- está definido justamente con la propiedad `transform`, y choca con el estilo ya definido si usamos la pseudoclase `:hover`. No hemos usado las propiedades `top`, `bottom`, `left` y `right` para posicionar los iconos con el posicionamiento absoluto definido para cada ícono, porque simplemente no funcionaban en este caso (solo funcionó correctamente la propiedad `transform`).

En segundo lugar, también es necesario decir que hemos ideado un sistema para conseguir que todos los **íconos de la sección IA de la landing-page sean fácilmente trasladables** como un bloque único si queremos hacer cambios de posición a posteriori. Esto es lo que consiguen las variables globales `-X` y `-Y` y las transformaciones lineales extra que hacemos con ellas en cada ícono que está alrededor del logo de Python, usando la función `calc()`. Véase más información de la lógica de todo esto en la figura 3.8:

```

/*[1]: Definimos posición absolute en relación al contenedor relativo dentro
de donde está el ícono (está dentro del #IA-bigdata > div:nth-child(6))*/

/*[2]: definimos translante mediante transform ajustando el centro a coordenadas --X e --Y de las
variables globales (que es el centro del círculo, que definimos como el 0,0
desde el cual construimos el posicionamiento de los iconos). Desde esa posición hacemos ajustes
para posicionarlo alrededor del círculo (para todos los iconos "satélite" a python) o para ajustar
variaciones de posición involuntarias producidas al cambiar width. Usar variables globales nos
permitirá mover todos los iconos de la sección IA alrededor del ícono de python -usaremos la misma
línea modificando las*/ 

/*[3]: Aquí no nos sirve box-shadow, debemos usar la propiedad filter para que envuelva
la imagen vectorial, no la caja que la contiene*/

/* logo Python */
#pythonLanding {
    position: absolute; /*[1]*/
    transform: translate(var(--x), var(--y));/*[2]: python centro del círculo*/
    width: 20em;
    filter: drop-shadow(7px 6px 2px □rgba(0, 0, 0, 0.25)); /*[3]*/
    transition: filter var(--tiempoTransicionEstandar) ease-in-out;
}

#pythonLanding:hover {filter: drop-shadow(7px 10px 2px □rgba(0, 0, 0, 0.25));}

/* logo TensorFlow (sombra hacia abajo) */
#tensorflowLanding {
    position: absolute;
    width: 19em;
    transform: translate(calc(var(--x) + .5em), calc(var(--y) + 34em)); /*logo tensorflow se posiciona sumando .5em a --X y 34em a --Y*/
    filter: drop-shadow(0px 10px 2px □rgba(0, 0, 0, 0.25));
    transition: filter var(--tiempoTransicionEstandar) ease-in-out;
}

#tensorflowLanding:hover {filter: drop-shadow(0px 14px 2px □rgba(0, 0, 0, 0.25));}

/* Scikit-Learn (sombra hacia arriba) */
#sklearnLanding {
    position: absolute;
    width: 15em;
    transform: translate(calc(var(--x) + 2em), calc(var(--y) - 23em));
    filter: drop-shadow(0px -7px 2px □rgba(0, 0, 0, 0.25));
    transition: filter var(--tiempoTransicionEstandar) ease-in-out;
}

#sklearnLanding:hover {filter: drop-shadow(0px -10px 2px □rgba(0, 0, 0, 0.25));}

/* Seaborn (sombra hacia arriba a la izquierda) */
#seabornLanding {
    position: absolute;
    width: 20em;
    transform: translate(calc(var(--x) - 25em), calc(var(--y) - 11em));
    filter: drop-shadow(-7px -7px 2px □rgba(0, 0, 0, 0.25));
    transition: filter var(--tiempoTransicionEstandar) ease-in-out;
}

#seabornLanding:hover {filter: drop-shadow(-10px -10px 2px □rgba(0, 0, 0, 0.25));}

```

Figura 3.8: **Posicionamiento de iconos** con CSS de la sección de tecnologías *ia* y *big-data* de la landing page. Se muestra el uso de la propiedad filter y de su animación con hover en la imagen de Python y algunos iconos de módulos o librerías de Python satélites al mismo (Tensorflow, Sklearn y Seaborn). Se muestra en [2] el uso de las variables CSS --X e --Y para definir un centro de coordenadas en (0,0) para el logo de Python. Luego, con el uso de transformaciones lineales de estas coordenadas mediante la función calc() (sumas y restas de “em”) conseguimos disponer los iconos de los distintos módulos para Python en un círculo que rodea el ícono de este lenguaje (en este caso la función calc se usa en el ícono de TensorFlow, sklearn y seaborn; pero hay 5 más en el CSS que no se muestran en la captura). NOTA: Obsérvese que el ícono de Python no tiene transformación lineal extra aplicada porque es el origen de las coordenadas, y por lo tanto, no usamos calc(), el de los demás, por el contrario, sí.

También es importante mencionar que se ha usado la propiedad `display` con `grid`, y `grid-template-columns` con `repeat(nVeces, 1fr)` para definir estructuras de columnas en varias secciones del index.html y el uso de `display` con `flex`. A saber:

- En la sección del contenedor onda usamos grid con 2 fractions o columnas que separan los dispositivos móviles de la presentación de la empresa, de forma clara.
- En la sección de las cards del front-end usamos grid con 3 fractions o columnas (e.g, ver figura 3.9).
- En la sección de las cards del back-end usamos grid con 2 fractions o columnas.
- En la footer se ha usado `display` con `flex`.

Figura 3.9: Uso de grid con tres columnas para generar tres columnas perfectamente espaciadas para la sección de front-end

```
/*Contenedor de las cards de la sección tecnologías front-end
| (CON GRID!)*/
#tec-front-end section:last-child {
    padding-top: 3.5em;
    max-width: var(--MaximaAnchuraCards);
    margin: 0 auto;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    text-align: center;      /*MANTENGO EL TEXTO CENTRADO DENTRO DE CADA CARD*/
    justify-items: center;  /* CENTRO LAS 3 CARDS HORIZONTALMENTE*/
}

}
```

NOTA: Reiterar que esta web no es un diseño responsive, porque no hemos usado media queries dado que no es el objetivo de la asignatura.

En relación a la **implementación de la interactividad**, la librería animate.style que contamos en el apartado de diseño tiene muy fácil implementación. Por ejemplo, para conseguir la animación del menú desplegable con tres elementos de la barra de navegación -en el index.html- solo ha sido necesario importar la librería animate.style mediante CDN en el head del html y definir dos propiedades CSS en el selector correspondiente (`animation` con el valor `fadeInDown` y `animation-duration` con el tiempo en segundos que queremos que tarde en terminar). Con el valor `fadeInDown` obtenemos el efecto de que los elementos de la navBar caen de arriba a abajo, y para que la animación sea secuencial (es decir, que cada elemento li de la lista de servicios aparezca primero antecediendo al siguiente en intervalos regulares) se ha combinado oportunamente el tiempo asignado a cada uno: el valor del siguiente es

el del anterior más una cantidad constante (el primero .8s, segundo 1.6s y tercero 2.4s) tal y como vemos en la figura 3.10.

Figura 3.10: En rojo el Código html (izquierda) y CSS (derecha) para conseguir la animación del menú desplegable con animate.style

```

<!-- SECCIÓN 0 (S0) -- HEADER: barra de navegación-->
<header>
  <nav>
    <!-- Aplicamos estilos para la barra de navegación -crucial añadir elemento
    <ul>
      <li>
        <span>Servicios</span> 
        <ul>
          <li><a href="#">Front-end</a></li>
          <li><a href="#">Back-end</a></li>
          <li><a href="#">IA & BigData</a></li>
        </ul>
      </li>
      <li><span><a href="#">Quiénes somos</a></span></li>
      <li><span><a href="#">Incorpórate</a></span></li>
      <li><span><a href="#">Contacto</a></span></li>
    </ul>
  </nav>
</header>

```

```

/*desplegable "Front-end"*/
nav ul li ul li:first-child {
  margin-right: 2.1em; /*[A]*/
  animation: fadeInDown; /*[B]*/
  animation-duration: .8s; /*[C]*/
}

/*desplegable "Back-end"*/
nav ul li ul li:nth-child(2) {
  margin-right: .65em; /*[A]*/
  animation: fadeInDown; /*[B]*/
  animation-duration: 1.6s; /*[C]*/
}

/*desplegable "IA y big data"*/
nav ul li ul li:last-child {
  margin-left: 3.6em; /*[A]*/
  animation: fadeInDown; /*[B]*/
  animation-duration: 2.4s; /*[C]*/
}

```

3.4. Mantenimiento

Hemos tomado una serie de medidas que hacen que la página sea más mantenible:

- Hemos usado **variables de color en la pseudoclase root**. Esto ayuda a poder hacer cambios rápidamente en la paleta de colores sin tener que acudir a sitios distintos del código, ni tener que cambiar varias veces el mismo color en distintos puntos.
- Hemos **evitado abusar de los ids y de las clases**, utilizando sobretodo selectores descendientes. Esto hace que sea más complicado que un usuario escriba un id o clase que ya existe y que los estilos choquen si se cambia el css. *Sin embargo*, a pesar que esto mantiene el código HTML más limpio, tiene un punto débil: si se desea cambiar la estructura de la página con HTML básicamente puede generar problemas (el que era el segundo hijo será el tercero si añadimos un elemento antes). Por lo tanto está encarado a no cambiar la estructura del DOM y solo a hacer variaciones de estilos.
- El index.html inicialmente lo habíamos codificado con varios archivos CSS customizados. Esto en producción al final lo hemos cambiado a un solo archivo. El motivo por el que lo hicimos así inicialmente es porque esto hace que si el usuario tiene claro donde ha definido cada parte, pueda localizar más rápidamente el código que busca: no se enfrenta a scrolls infinitos, por un lado; y puede colaborar en GitHub con su compañero con menos conflictos porque la probabilidad de estar todos picando código en las mismas líneas del mismo

archivo CSS se reduce. Sin embargo, **en la versión final entregada se ha dejado en un solo archivo CSS** porque sino posiblemente sería un infierno para las personas que se encuentren con esta página a futuro y no trabajen con la misma metodología.

Ampliaciones

4.1. Mejora de código

El código posiblemente se podría mejorar con un preprocesador de CSS, tal como SCSS. Este lenguaje compila archivos CSS y permite escribir los selectores CSS agrupados de forma jerárquica. Sin embargo, la web no se ha hecho así dado que uno de los requerimientos funcionales era desarrollar la web con CSS puro.

Estaría bien encontrar una forma de posicionar los iconos de la sección IA sin usar la propiedad *transform*, y tomando solo las propiedades *top*, *right*, *bottom*, *left* del posicionamiento absoluto. De este modo podríamos animar también la propiedad *transform* de los mismos, sin problemas, usando `:hover`.

Finalmente, mencionar que ha habido algo que nos ha parecido misterioso. Un problema con el código que hemos podido solventar con un “parche”, pero que no hemos encontrado el por qué se ha producido. Si nos fijamos en la figura 3.10 veremos que hemos necesitado definir un `margin-left` o `margin-right` para cada elemento `li` del menú desplegable que contiene los servicios en la *navBar*. Sin esos margins como parche, los elementos de la lista desplegable no hubiesen quedado alineados verticalmente a pesar de usar las propiedades pertinentes para hacerlo. Seguro que tiene una explicación elegante, pero por ahora, no la hemos encontrado y el parche de definir los márgenes ha sido la mejor baza que hemos encontrado para centrarlos.

4.2. Soporte de idiomas

No se prevén hacer distintos idiomas para esta página web.

Evaluación y Conclusiones Finales

Hemos utilizado todas las herramientas a nuestra disposición para hacer una página agradable a la vista, usable y minimalista. Creemos que como primera aproximación a la creación de interfaces nos hemos encaminado correctamente, tomando ideas prestadas de los grandes jugadores en este campo y usando las herramientas colaborativas (GitHub, Figma, Trello) que hoy día ya empiezan a resultarnos indispensables.

Continuaremos el proyecto intentando definir las páginas internas con la misma metodología que hemos seguido hasta ahora, si el feedback es positivo; y aquellos aspectos que sean susceptibles de mejora serán cambiados.

ANEXO: memoria con Figma

En las siguientes secciones mostramos la maquetación de la landing page (de la que tenemos dos maquetaciones alternativas):

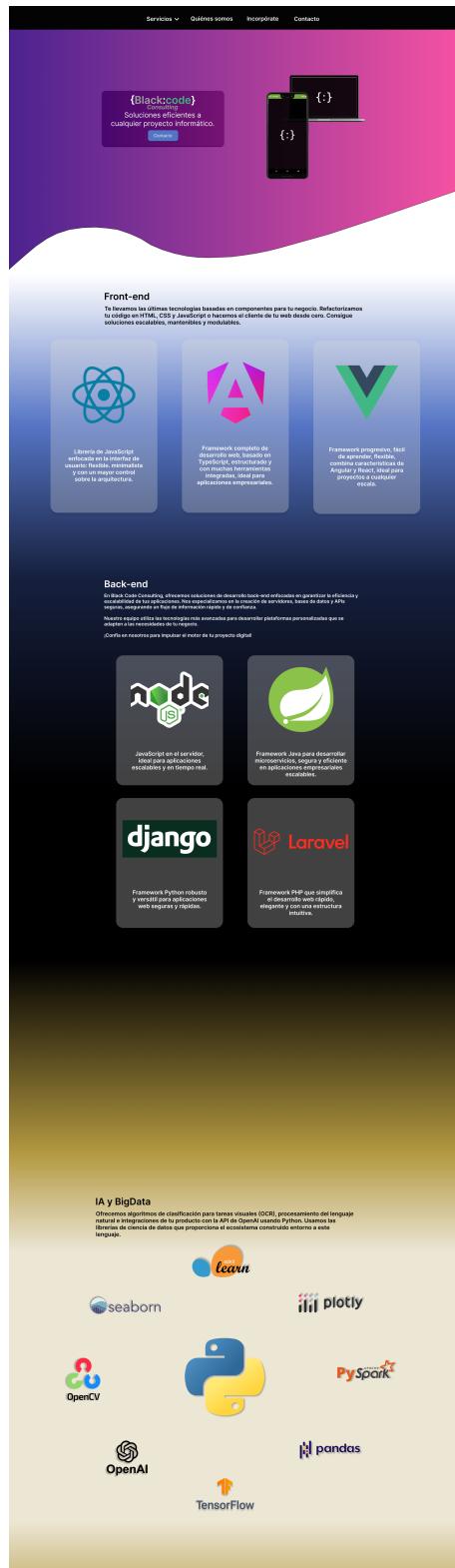
- **Landing Page (con dispositivos)** (sección 6.1): en esta versión -la programada finalmente en index- mostramos el “contenedor onda” con la presentación de la empresa y unos dispositivos con el logo simplificado de la misma.
- **Landing Page (sin dispositivos)** (sección 6.2): en esta otra versión el “contenedor onda” no tiene los dispositivos. Estudiamos la posibilidad de sustituirlo por el primero más adelante.

También mostramos las seis páginas internas de la web, a saber:

- **Front-end** (sección 6.3).
- **Back-end** (sección 6.4).
- **IA y BigData** (sección 6.5).
- **Incorpórate** (sección 6.6).
- **Quienes somos** (sección 6.7).
- **Contacto** (sección 6.8).

Además, por si el visionado se complica por la limitación de anchura de las páginas más largas, todos los *.png* del *mockup* pueden encontrarse también en este link, en nuestro repositorio de github, carpeta [mockupFigma](#).

6.1. Landing Page (index.html) con dispositivos



6.2. Landing Page (index.html) sin dispositivos



6.3. Front-end (página interna)

The screenshot shows a landing page for "Front-End development". At the top, there's a navigation bar with links: Servicios, Ofertas, Portfolio, Contacto. Below the header, the title "Front-End development" is displayed.

¿En qué desarrollamos y por qué?

Desarrollamos el cliente en vue, angular y react. Son los tres frameworks de front-end más codificados y aceptados en el mercado, y te pueden proporcionar las siguientes ventajas:

- Aplicaciones escalables por componentes**: Consigue hacer crecer la base de datos de forma modular y segura. Idónea para aplicaciones grandes o que tienen previsto de crecer rápidamente.
- interfaces dinámicas y reactivas**: Actualizan automáticamente el cliente de los cambios en los recursos o cambios de datos sin recargar toda la página. Mejora la velocidad de carga y la experiencia de usuario.
- Comunidad y soporte**: Tienen una comunidad de desarrolladores que facilita encontrar soluciones a problemas frecuentes.

¿Qué diferencias fundamentales hay entre los tres frameworks?

The comparison section features a central image of the Vue.js logo (a green triangle) with arrows pointing left and right, set against a dark background with code snippets on either side. Below this are four comparison boxes:

- Suave curvo de aprendizaje**: Vue permite aprender la mayoría de su lógica para manejar el framework. Para manejar VUE y tener recursos limitados es un framework ideal.
- Integración gradual**: A diferencia de angular o react, vue se puede integrar en proyectos existentes sin modificar la totalidad del proyecto.
- Sistema de enlace de datos reactivo**: En este sistema, cualquier cambio en los datos del modelo se refleja en la interfaz de usuario de forma inmediata sin necesidad de intervención manual.
- Ecosistema de datos liviano pero extensible**: A diferencia de angular, vue es de un ecosistema liviano que da la libertad de usar otras tecnologías tales como (internacionalización, etc.) conforme tu proyecto crezca.

Característica	Vue.js	React.js	Angular
Curva de aprendizaje amigable	o		
Integración gradual (progresivo)	o	o	
Sistema de enlace de datos reactivo	o		
Ecosistema liviano pero extensible	o	o	
Virtual DOM	o		
Uso de JSX para interfaces complejas	o		
Alto rendimiento en interfaces grandes	o		
Ecosistema amplio (React Native, etc.)	o		
Flexibilidad total en elección de herramientas	o		
Framework completo con herramientas integradas		o	
Basado en TypeScript		o	
Inyección de dependencias		o	
Patrón MVVM (Model-View-ViewModel)		o	
Estructura robusta y escalable		o	
Ideal para grandes aplicaciones empresariales		o	

At the bottom of the page, there's a footer with links: Servicios (Front-end Development, Back-end Development, UI/UX Design), Empresa (Quiénes somos, Innovación), and Contacto (info@decodicoding.com, Formularios). A copyright notice at the very bottom states: © 2022 | Todos los derechos reservados.

6.4. Back-end (página interna)

Servicios ▾ Quiénes somos Incorporarse Contacto

Back-End Development



Black Code Consulting: Innovación y Eficiencia en Desarrollo Backend

En Black Code Consulting, somos expertos en el desarrollo backend utilizando tecnologías de vanguardia como Node.js, Django, Spring Boot y Laravel. Nuestro objetivo es construir sistemas robustos, eficientes y escalables que propulsen a nuestros clientes hacia el éxito en un entorno digital en constante evolución.

Por qué Usamos Estas Tecnologías

Nuestra elección de estas tecnologías se basa en su capacidad para adaptarse a diferentes necesidades de los proyectos y en su reputación en la industria por proporcionar soluciones efectivas y escalables. Esto nos permite abordar proyectos que van desde aplicaciones simples hasta sistemas complejos y de gran escala, asegurando que nuestros clientes siempre reciban lo mejor en términos de rendimiento y confiabilidad.

¿Qué Hacemos?

- Servidores de Alto Rendimiento: Diseñamos e implementamos servidores que garantizan un rendimiento óptimo, incluso bajo altos volúmenes de tráfico y datos.
- Gestión de Bases de Datos Complejas: Nuestro equipo se especializa en la gestión y optimización de bases de datos, asegurando un acceso rápido y eficiente a la información.
- Desarrollo de APIs Rápidas y Seguras: Creamos APIs personalizadas que no solo son rápidas, sino también seguras, facilitando la integración de sistemas y mejorando la experiencia del usuario.

Nuestro Enfoque Personalizado

Entendemos que cada cliente es único. Por eso, trabajamos de la mano con nuestros clientes para comprender sus necesidades específicas y adaptar nuestras soluciones a sus objetivos comerciales. Ya sea desarrollando aplicaciones en tiempo real con Node.js, construyendo sistemas empresariales con Spring Boot, o creando proyectos web ágiles con Django y Laravel, estamos listos para enfrentar cualquier desafío.



Casos de Éxito

En Black Code Consulting, estamos comprometidos con la excelencia y la satisfacción del cliente.

A lo largo de nuestra trayectoria, hemos tenido el privilegio de colaborar con diversas empresas, ayudándolas a alcanzar sus objetivos a través de soluciones innovadoras y personalizadas. A continuación, presentamos algunos de nuestros casos de éxito, que reflejan cómo hemos transformado desafíos en oportunidades y generado resultados significativos.

New Logistik:
Implementamos un sistema de gestión de datos utilizando Django, optimizando sus operaciones logísticas y reduciendo el tiempo de procesamiento en un 50%.
Esta solución proporcionó una mejor visibilidad de la cadena de suministro y una respuesta más ágil a las demandas del mercado.

ABC Fintech:
Desarrollamos una API segura y escalable con Node.js, que facilitó la integración de múltiples servicios financieros. Gracias a esta solución, ABC Fintech pudo lanzar nuevas funcionalidades en un 30% menos de tiempo, mejorando su eficiencia operativa en un mercado dinámico.

Flow Systems:
Creamos una plataforma personalizada de gestión de proyectos utilizando Laravel, que integró herramientas de colaboración en tiempo real. Como resultado, Flow Systems experimentó un aumento del 40% en la productividad y una notable mejora en la satisfacción del cliente.

Servicios
Front-end Development
Back-end Development
IA + Big Data

Empresa
Quiénes somos
Incorporate

Contacto
info@blackcodeconsulting.com
Formulario 

© 2024 | Todos los derechos reservados

6.5. IA y BigData (página interna)



6.6. Incorporárate



¿Tienes Talento?

Detrás de todo gran cambio siempre hay una gran persona. Cada día nuestros profesionales hacen cosas increíbles al trabajar juntos para conseguir nuestro propósito: cumplir la promesa de la tecnología y el ingenio humano.

Ven a formar parte de nuestro equipo – aporta tus ideas, tu ingenio y tu determinación para marcar la diferencia, y resolveremos algunos de los mayores retos del mundo.

Trabajamos con personas excepcionales, con la tecnología más puntera y con empresas líderes en todos los sectores para crear valor para nuestros clientes, personas y comunidades.

Elige una carrera con nosotros y juntos crearemos un valor positivo y duradero.

Únete al equipo

Junior Backend Developer / Java

Ubicación: Híbrido (CDMX)
Descripción: Colabora en el desarrollo y mantenimiento de nuestras aplicaciones backend.
Requisitos: Conocimientos en Java, SQL, y API REST.

IA Developer / Python

Ubicación: Remoto
Descripción: Apoya en el desarrollo de modelos de inteligencia artificial y procesamiento de datos.
Requisitos: Conocimientos básicos en Python y Machine Learning.

Full Stack Developer / Angular

Ubicación: Remoto
Descripción: Participa en el desarrollo de aplicaciones web con Angular en el frontend y Node.js en el backend.
Requisitos: Experiencia con Angular, Node.js y manejo de APIs REST.

Servicios

Front-end Development
Back-end Development
IA + Big Data

Empresa

Quiénes somos
Incorporate

Contacto

info@blackcodeconsulting.com
Formulario



6.7. Quiénes Somos



Sobre nuestra empresa



Nuestra Historia:

En Black-Code Consulting, comenzamos con una visión clara: crear soluciones tecnológicas que marcaran la diferencia en un mundo digital en constante evolución.

Desde nuestros primeros proyectos, hemos crecido junto a nuestros clientes, desarrollando herramientas y plataformas innovadoras que impulsan sus negocios.

En Black-Code Consulting, nuestra empresa y nuestra cultura se parecen mucho a nuestro producto. Están conectadas, no ensambladas, para lograr una excelente experiencia.

Nuestra misión: Ayudar a millones de empresas a crecer mejor

Creemos que no sólo es importante crecer más, sino crecer mejor. Y esto significa garantizar el éxito de tu empresa y también el de tus clientes. Todo el mundo gana.



Nuestro Equipo

En Black-Code Consulting, contamos con un equipo diverso y altamente capacitado que trabaja en conjunto para ofrecer soluciones innovadoras y de calidad.

Cada miembro aporta su experiencia y pasión, y juntos nos comprometemos a alcanzar la excelencia, impulsando el éxito de nuestros clientes y el crecimiento continuo de nuestra empresa.



Sostenibilidad

Nos comprometemos con la sostenibilidad en cada paso que damos. Implementamos prácticas responsables que minimizan el impacto ambiental, desde la optimización de nuestros procesos hasta el uso de tecnología eficiente.

Creemos en el poder de la innovación para construir un futuro más limpio, y trabajamos constantemente para integrar soluciones sostenibles que contribuyan al bienestar de las personas y el planeta.

Localización y Entorno

Nuestra sede se encuentra Valencia, España, una ciudad vibrante y en constante crecimiento tecnológico. Ubicada en la costa mediterránea, Valencia es conocida por su innovación y calidad de vida. Nuestra sede se beneficia de un entorno dinámico que combina cultura, sostenibilidad y un ecosistema empresarial en expansión.

Carrer D'Alberic, 18, Extramurs, 46008
València



Servicios

Front-end Development

Back-end Development

IA + Big Data

Empresa

Quiénes somos

Incorpórate

Contacto

info@blackcodeconsulting.com

Formulario

© 2024 | Todos los derechos reservados

6.8. Contacto

The screenshot shows a website's contact page. At the top, there is a black navigation bar with white text: "Servicios ▾", "Quiénes somos", "Incorpórate", and "Contacto". Below this is a large, semi-transparent purple-to-blue gradient banner. In the center of the banner, the word "Contáctanos" is written in a large, white, sans-serif font. To the right of the banner is a white contact form box with rounded corners. It contains three input fields with labels: "Tu nombre*" above a text input field, "E-mail*" above another text input field, and "Comentarios:" above a larger text area. At the bottom of the form is a blue rectangular button with the word "Contacto" in white. At the very bottom of the page is a dark footer section divided into three columns by vertical lines. The left column is labeled "Servicios" and lists "Front-end Development", "Back-end Development", and "IA + Big Data". The middle column is labeled "Empresa" and lists "Quiénes somos" and "Incorpórate". The right column is labeled "Contacto" and lists the email "info@blackcodeconsulting.com", a link to a "Formulario", and a small blue LinkedIn icon. A thin horizontal line separates the footer from the rest of the page.

Servicios

Front-end Development

Back-end Development

IA + Big Data

Empresa

Quiénes somos

Incorpórate

Contacto

info@blackcodeconsulting.com

Formulario

© 2024 | Todos los derechos reservados

Bibliografía

- [1] Andrew J. Elliot. Color and psychological functioning: a review of theoretical and empirical work. *Frontiers in Psychology*, 6:368, Apr 2015. Accessed: 2024-11-10, Available at: <https://PMC4383ss146>.