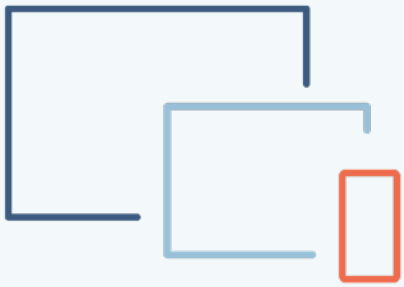
Universitat Oberta de Catalunya



Administración Digital Fácil

Administración Digital Fácil

Tus trámites en un único lugar

Memoria de Proyecto Final de Máster

Máster Universitario en Desarrollo de sitios y aplicaciones web

Autor: Sara Esteban Miguel

Tutor de TF: Carles Arnal Castello

Profesor: César Pablo Córcoles Briongos

FICHA DEL TRABAJO FINAL

Título del trabajo:	Administración Digital Fácil
Nombre del autor:	Sara Esteban Miguel
Nombre del consultor:	Carles Arnal Castello
Nombre del profesor:	César Pablo Córcoles Briongos
Fecha de entrega:	11/2022
Titulación:	MU Desarrollo de sitios y aplicaciones web
Área del Trabajo Final	Desarrollo de aplicaciones web
Idioma del trabajo:	Castellano
Palabras clave:	Administración electrónica, brecha digital, TFM, aplicación web
Keywords:	e-government, digital gap, Master's thesis, web app

Resumen del Trabajo:

El objeto del presente trabajo constituye la realización de un proyecto de cierre en la formación del máster universitario en Desarrollo de Sitios y Aplicaciones Web mediante el desarrollo de una aplicación web que englobe una parte de las herramientas y conocimientos adquiridos a lo largo del itinerario.

Administración Digital Fácil es una aplicación web que buscar acercar la administración electrónica de un modo sencillo e intuitivo para la ciudadanía, consiguiendo así, ayudar a la comunicación entre los diferentes organismos públicos y la población independientemente de su destreza digital.

El proyecto se realiza en forma de aplicación web para poder ser utilizada en cualquier dispositivo sin tener que contar con una adaptación específica dependiente de sistema operativo o tamaño del terminal. El diseño perseguirá ser responsivo e intuitivo.

El desarrollo para llevar a cabo este trabajo final se realizará, como herramienta principal, con el framework de Angular. Al mismo tiempo se intentará aprender una nueva tecnología desde cero a lo largo de este recorrido, incorporando la conexión del backend con Nest.

Abstract:

The main goal of this work is to carry out a closure project in the education leading to a master's degree in Web App and Website Development through the development of a web application where most of the tools and knowledge acquired can be included.

Administración digital fácil is a web application whose main axis seeks to bring electronic administration closer to citizens in a simple and intuitive way, thus helping communication between the different public organismos and the population regardless of their digital skills.

The project is carried out in the form of a web application that can be used on any device without having to rely on a specific adaptation depending on the operating system or the size of the terminal. The design will aim to be responsive and intuitive.

The development to carry out this final work will be done, as the main tool, with the Angular framework. At the same time, it will be attempted to learn a new technology from the ground up while incorporating the backend connection with Nest.

AGRADECIMIENTOS

A mi hermano por su apoyo incondicional y cuidarme siempre.

A mi familia por aguantar la presión que ha supuesto compatibilizar este máster con la vida.

A Marta por todos los aportes al proyecto y alentarme cuando quería rendirme.

A los médicos que hicieron posible que la pandemia no dejase incompleto este máster.

A mi tutor de la UOC, Jordi Gervas, porque no ha sido el recorrido más sencillo y siempre ha encontrado soluciones.

A los consultores de cada una de las asignaturas, cuyas enseñanzas concluyen en este proyecto.

Al tutor del trabajo final, Carles Arnal, por sus aportes y resolver todas las dudas.

A todos los que han hecho simple mi caos, ¡Gracias!

DEDICATORIA

A mi madre, a la que le sigo debiendo un "curso de ordenador"
y que sigue intentando "modernizarse" todos los días.

A Nieves y Eduardo, a quienes hace años me enseñaron
que la tecnología no tiene que ver con la edad.

CITAS

"*Life is really simple, but we insist on making it complicated*"

Confucio
Filósofo

"*A little simplification would be the first step toward rational living, I think*"

Eleanor Roosevelt
Escritora
Activista
Política

"*Clutter and confusion are failures of design, not attributes of information*"

Edward Tufte
Estadístico
Profesor

"*Don't make the process harder than it is*"

Jack Welch
Empresario
Ingeniero

"*Everything should be made as simple as possible, but not simpler*"

Albert Einstein
Físico

ÍNDICE

ÍNDICE DE FIGURAS (y tablas)

1. Introducción

1.1. Contexto y justificación del trabajo

La Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos expone en sus motivos iniciales que "el tiempo actual -y en todo caso el siglo XXI, junto con los años finales del XX-, tiene como uno de sus rasgos característicos la revolución que han supuesto las comunicaciones electrónicas", dando así inicio al inicio de una legislación estatal que pretendía "estar a la altura de los tiempos" acompañando y promoviendo el uso de las comunicaciones electrónicas en beneficio de los ciudadanos.

Tal y como analizaré a continuación, son varios los momentos en los que la Unión Europea y España han decidido legislar a favor de las nuevas tecnologías con el objetivo de facilitar el acceso a los ciudadanos. El problema, como también concluiremos más tarde, es que, a pesar de las buenas intenciones, en muchas ocasiones se olvidan de sectores de la sociedad vulnerables, que quedan excluidos en muchas ocasiones. Además, debemos analizar que, circunstancias próximas a las administraciones más cercanas al ciudadano y acontecimientos de salud global, han llevado a realizar parte del proceso de una forma bastante abrupta, que, confiemos, sea mejorada en el futuro.

Consejo Europeo de Santa María da Feira 2000

Podríamos fijar el inicio del impulso de la administración electrónica en la cumbre europea que se celebró en junio de 2000 en Portugal. Tal y como podemos comprobar en las Conclusiones de la Presidencia, fueron varias las líneas de actuación en las que se instaba a los poderes públicos comunitarios en aumentar la comunicación electrónica.

En concreto podemos fijarnos en las líneas "5. Mejorar el acceso a la línea", "6. Sacar más provecho del Mercado Único", "9. Modelos de empresa electrónica próspera y apoyo de alto nivel a la pequeña empresa".

En esta reunión se apostaba por la vía electrónica para favorecer en especial a la pequeña empresa comunitaria. Sin embargo sirvió para que muchos países, entre otros España, contemplasen la posibilidad de ampliar el público objetivo de las nuevas medidas.

Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos

Como consecuencia de los compromisos alcanzados 7 años antes en Portugal, ve la luz una Ley cuyo objeto se define como "el derecho de los ciudadanos a relacionarse con las Administraciones Públicas por medios electrónicos y regular los aspectos básicos de la utilización de las tecnologías de la información en la actividad administrativa [...]"

También es cierto que dentro de los fines de la Ley podemos encontrar el de "facilitar el acceso por medios electrónicos de los ciudadanos a la información y al procedimiento administrativo, con especial atención a la eliminación de las barreras que limiten dicho acceso". Algo que podríamos entender como crear un sistema de comunicación accesible, más si tenemos en cuenta que también aboga por la simplificación de procedimientos, principio de igualdad y principio de accesibilidad. Algo que se concreta aún más en el Artículo 8 de Garantía de prestación de servicios y disposición de medios e instrumentos electrónicos.

También se aboga por la Formación de empleados públicos en su Disposición adicional segunda, aunque solo lo hace como una elección para administraciones locales, provinciales y regionales.

Como curiosidad, cuenta con más de dos hojas finales en un anexo para definir el vocabulario empleado a lo largo de la Ley. Conceptos como "Aplicación". "Dirección electrónica" o "Firma electrónica" parecen requerir de explicación más allá de su uso en la normativa.

Esta Ley debía entrar en vigor al día siguiente de su publicación, sin embargo muchas de las propuestas requerían de modificaciones de otras leyes y plazos que hacen que 15 años después no todas las Administraciones cuenten con esas vías de comunicación tan accesibles para el ciudadano.

Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas

Ocho años después de la primera ley sobre cuestiones administrativas electrónicas, se revisa la normativa vigente dando como resultado una nueva Ley sobre el Procedimiento Administrativo Común, y en su artículo 14 define quiénes tienen derecho y obligación a relacionarse electrónicamente con las Administraciones Públicas.

Si bien reconoce la libre elección para todas las personas físicas, obliga a esta comunicación telemática a las personas jurídicas, entidades sin personalidad jurídica, profesionales colegiados, representantes de los obligados y empleados públicos. Y no hace ningún tipo de distinción, lo que fuerza a cualquier colectivo o asociación, con cualquier condición de capacidad o conocimiento técnico a constituirse y mantener comunicación electrónica.

Resulta significativo el hecho de que en esta Ley ya no se encuentra ningún tipo de propuesta a la ventanilla única y en cambio se refuerza, a lo largo de todos los procedimientos, la necesidad de sedes electrónicas para cada una de las administraciones. Tampoco impulsa, ni recomienda, ninguna formación para los trabajadores públicos que puedan ayudar a los ciudadanos y colectivos finales.

Al tratarse de una Ley de amplia dimensión, se da un plazo de un año para su entrada en vigor, sin embargo se conceden seis años para poner en marcha todas las cuestiones que

llevan a la administración electrónica, fijando los efectos en el 2 de abril de 2021, en pleno confinamiento por la crisis sanitaria global de la covid 19.

Brecha y analfabetismo digital en España 2015-2020

Analfabetismo digital: nivel de desconocimiento de las nuevas tecnologías, que impide a las personas el acceso a las posibilidades de interactuar con las mismas. Es decir, personas que no tienen capacidad para navegar por internet, acceder a contenidos multimedia, usar redes sociales, crear documentación electrónica o diferencias información relevante de superflua a través de medios digitales.

Brecha digital: cualquier distribución desigual en el acceso, uso o impacto de las tecnologías de la información y la comunicación (TIC) entre grupos sociales.

España no se caracteriza por liderar una gran formación en las TIC dentro de Europa. En 2015, UGT ya presentaba unas conclusiones que debían hacer replantear algunas cosas en su informe "La brecha digital en España", y es que el 47% de los españoles contaba con un nivel de formación en TIC muy bajo o inexistente.

Aunque en los últimos años, el porcentaje de hogares sin conexión de calidad a internet a disminuido, y casi la totalidad de viviendas cuentan al menos con un dispositivo capaz de conectarse, los datos del INE referentes al año 2020 mostraban conclusiones que hacen pensar en una segunda brecha digital en la que el acceso a internet y los equipos tecnológicos no son el problema, frente a la capacidad de los usuarios de manejarse.

En el año de la pandemia, un 22% de españoles no usaron internet, un 24% no usaron el correo electrónico, un 38% no usaron la banca electrónica, y un 35% reconoce no tener los conocimientos necesarios para comunicarse con las Administraciones Públicas.

Estos datos son mayores entre las personas más vulnerables. Aquellos que viven en situación de pobreza, desamparados o con capacidades cognitivas y/o motoras diversas.

Proyecto propuesto

Con los datos anteriormente expuestos podemos concluir que un amplio porcentaje de la población no podría constituirse como forma jurídica y comunicarse con la administración correspondiente si no es con la participación de un tercero.

Todo toma más importancia cuando observamos que existen administraciones, especialmente locales, que han implantado sus sistemas digitales casi en extremis, en momentos pandémicos y que solo han incorporado sus trámites físicos a un formato electrónico, sin simplificación alguna.

Además, el ciudadano de a pie, incluso con recursos digitales, muchas veces es incapaz de diferencias qué administración es la encargada de realizar los trámites que necesita, ya que todas ellas pueden dar subvenciones, registrar, validar…

Solo es necesario echar un vistazo a sedes electrónicas locales, por ejemplo la de Ayuntamiento de Aranda de Duero, que aglutina todos los trámites obligando al usuario a saber exactamente quién tramita y con unas explicaciones extremadamente largas y complejas de entender. Algo que se ve mejorado en algunas administraciones locales más grandes como el Ayuntamiento de Zaragoza.

Así, no resulta extraño que muchas de las asociaciones de colectivos más vulnerables se vean abocadas a la extinción o pérdida de algunas prestaciones por las barreras que supone la comunicación con las entidades correspondientes.

La propuesta de solución al problema es la creación de una aplicación web que, dependiendo de un ayuntamiento, pueda ofertar un catálogo amplio, sencillo y accesible de todos los trámites locales, regionales y estatales, primando siempre la lectura fácil. Teniendo como referente la idea que desarrollo el Ayuntamiento de León para el apartado del catálogo de trámites de su proyecto inclusivo PCOS.

En la propuesta tendríamos un diseño de página que utilizase cualquier usuario con los requisitos mínimos para llegar hasta la página, y un cms para la gestión de la información por parte de los trabajadores de la administración correspondiente, quienes dependiendo de su condición de usuario registrados y privilegios, tendrán un acceso u otro.

La aplicación se desarrollará como una SPA con ayuda de Angular. Los datos locales elegidos serán los del Ayuntamiento de Aranda de Duero, para tratar de crear una mejor arquitectura de la información.

1.2. Objetivos del trabajo

Este proyecto, como trabajo fin de máster, persigue poner en práctica la mayor parte de los conocimientos adquiridos a lo largo de la formación del mismo, así como el aprendizaje de nuevas herramientas.

Objetivo principal

- Desarrollo, en Angular, de una aplicación web desde la idea hasta su versión operativa en línea en la que se planifiquen y desarrollen todas sus etapas, para dar solución a un problema planteado y analizado.

- Poner en práctica los conocimientos y habilidades adquiridos a lo largo del máster, además de las herramientas para seguir ampliando nuevo conocimiento y técnicas de aprendizaje que puedan surgir en el futuro de la rama elegida como área formativa.

Objetivos secundarios

- Analizar y planificar una solución en base a un problema.
- Gestión eficaz del proyecto, con los correspondientes ajustes para concluir la tarea con éxito. Basándose en los principios ágiles
- Diseño de interfaces de acuerdo a los principios de usabilidad, recordando que las aplicaciones tienen que tener como finalidad última su empleo por parte del usuario final y no la demostración de complicaciones del desarrollador.
- Ampliar el conocimiento de las tecnologías de desarrollo web, en especial Angular, Nest y TypeScript, mediante el desarrollo de una API REST con sus correspondientes endpoints que puedan ser consumidos por la aplicación desarrollada para el frontend, así como suficientemente comentada para poderse reutilizar con otros proyectos y desarrolladores.
- Aprender y utilizar el manejo de flujos modernos de trabajo utilizando herramientas como Git, Node, Docker, Postman…

1.3. Impacto en sostenibilidad, ético-social y de diversidad

El desarrollo de este trabajo trata de obtener un impacto positivo dentro de las competencias de comportamiento ético y global. De hecho podemos identificar algunas de las metas deseadas en varios de los Objetivos de Desarrollo Sostenibles propuestos por la ONU.

Dimensión de sostenibilidad

Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.

En su meta 9.4 propone: "De aquí a 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías [...]"

La Administración, tanto a nivel nacional como europeo, busca desde hace años adoptar las medidas tecnológicas a su alcance. Por lo que desarrollar una herramienta que facilite esa implantación en la ciudadanía podría considerarse como un **impacto positivo**.

Por otro lado, cabe destacar, que el hecho de que una aplicación similar fuese recomendada para cada Administración, aumenta significativamente el número de recursos electrónicos utilizados, lo que tendría un **impacto negativo** en la generación de huella de carbono si se pusiera en marcha en cada uno de los organismos locales, provinciales, autonómicos, estatales… que podrían verse reducidas si solo se implantase la tan ansiada ventanilla única.

Dimensión comportamiento ético y de responsabilidad social (RS)

Objetivo 1: Poner fin a la pobreza en todas sus formas en todo el mundo.

Objetivo 8: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos.

Objetivo 16: Promover sociedades justas, pacíficas e inclusivas.

En todos estos objetivos de responsabilidad social encontramos metas que pueden verse beneficiadas mediante la realización de proyectos que ayuden a simplificar el acceso electrónico en las comunicaciones con la administración:

> 1.4: Garantizar los mismos derechos en las nuevas tecnologías entre otras materias, para erradicar la pobreza.
>
> 8.5: Lograr el empleo pleno y productivo, solo se puede conseguir eliminando la barrera electrónica para todos esos profesionales sin altas capacidades digitales que están obligados a relacionarse por este medio con la Administración.
>
> 16.b: Promover y aplicar leyes y políticas no discriminatorias, en las que la barrera tecnológica no suponga un desistimiento.

En un mundo digital, en el que el desarrollo profesional debe ir de la mano de los avances tecnológicos, cualquier herramienta, por pequeña que sea que reúna a todas las partes sin distinción de conocimientos, podemos valorarlo desde un punto de vista de **impacto positivo**.

Dimensión diversidad, género y derechos humanos

Objetivo 10: Reducción de las desigualdades.

> 10.2: Potenciar y promover la inclusión social, económica y política de todas las personas, independientemente de la edad, sexo, discapacidad, raza, etnia, religión o situación económica u otra condición.
>
> 10.3: Garantizar la igualdad de oportunidades y reducir la desigualdad de resultados […]

Son suficientes estas dos metas para ser conscientes del camino que queda por realizar para alcanzar una igualdad real. Son muchas las personas que bien por sus diversas capacidades de comprensión, bien por su conocimiento formativo en la materia, tienen grandes dificultades para establecer comunicación con la Administración y en ocasiones dejan pasar oportunidades por este motivo.

La aplicación propuesta tiene como prioridad mejorar este apartado, con redacciones de lectura fácil para poder alcanzar a un mayor número de personas, y generar un **impacto positivo**.

Respecto al resto de dimensiones de este apartado, **no existe un impacto directo** en cuestiones como la perspectiva de género, ya que la solución propuesta valora únicamente el acceso de la ciudadanía sin diferenciar entre géneros, razas, religiones, orientaciones…

1.4. Enfoque y método seguido

El desarrollo web, en muchas ocasiones, suele ser un trabajo de equipo donde diversos perfiles profesionales entrelazan las diferentes áreas de diseño, desarrollo y contenido. Las metodologías ágiles son las más empleadas en los últimos tiempos para estas tareas.

En el caso de Administración Digital Fácil nos enfrentamos a un proyecto unipersonal, que además debe compartir el tiempo de trabajo disponible con un empleo a jornada completa y el aprendizaje de nuevas tecnologías, por lo que es todavía más importante implementar una metodología efectiva y eficaz para dedicar el mayor tiempo posible al proyecto. Razón por la que se ha decidido emplear una metodología similar a la de un equipo; en este caso la elección son los principios de la metodología Kanban, teniendo en cuenta que todas las fases del proceso están interconectadas y se desarrollarán casi en paralelo, afectando unos procesos a otros constantemente.

Metodología Kanban

Se trata de un método de gestión de trabajo surgido en Toyota Production System a finales de los años 40. Su propósito fundamental consiste en minimizar los desperdicios sin afectar a la producción, de manera que se cree más valorar sin generar más gasto.

Los principios de la metodología, y que la diferencian de las otras ágiles son:

- Calidad garantizada: todo tiene que salir bien a la primera, no hay margen de error. Kanban no premia la rapidez, sino la calidad.

- Reducción del desperdicio: se trata de hacer solamente lo necesario, y hacerlo bien. Lo que supone la reducción de todo lo superficial y secundario.

- Mejora continua: además de la gestión, es un sistema de mejora en el desarrollo de los proyectos, dependiendo de los objetivos a alcanzar.

- Flexibilidad: la siguiente tarea a realizar se decide en el backlog (o tareas pendientes), pudiendo, de esta manera, priorizar las tareas entrantes según las necesidades del momento, dando una respuesta inmediata a las tareas imprevistas e importantes.

Metodología Kanban aplicada a Administración Digital Fácil

Para llevar a cabo esta metodología en el proyecto se ha creado el tablero básico de Kanban con las columnas de "Tareas por hacer", "En curso" y "Finalizada" por las que pasarán todas las tareas de los distintos hitos. Para llevarlo a cabo se utilizará Notion.

1.5. Planificación del trabajo

A continuación se muestra la planificación del proyecto, con los hitos más importantes y las tareas que engloban cada uno de ellos.

Etapas del proyecto	Inicio	Final	Cumplimiento
Entregas	**28/09/2022**	**16/01/2023**	
PEC 1	28/09/2022	11/10/2022	Entregada
PEC 2	12/10/2022	09/11/2022	Entregada
PEC 3	10/11/2022	18/12/2022	Entregada
Entrega final	19/12/2022	16/01/2023	Entregada
Gestión del proyecto	**21/02/18**	**10/06/18**	
Idea	28/09/2022	30/09/2022	Completado
Planificación	01/10/2022	11/10/2022	Completado
Seguimiento y control	01/10/2022	16/01/2023	Completado
Documentación	**01/10/2022**	**16/01/2023**	
Memoria del proyecto	01/10/2022	16/01/2023	Completado
Fuentes de información	01/10/2022	16/01/2023	Completado
Listado de funcionalidades	01/10/2022	11/10/2022	Completado
Análisis de mercado	12/10/2022	09/11/2022	Completado
Viabilidad	12/10/2022	09/11/2022	Completado
Encuesta	16/10/2022	16/12/2022	Completado
Preparación de datos	16/10/2022	16/01/2023	Completado
Presentación de primera versión	10/11/2022	18/12/2022	Retrasado: 16/01/2023

1.1. Tabla de etapas del proyecto

Etapas del proyecto	Inicio	Final	Cumplimiento
Vídeo de defensa	19/12/2022	16/01/2023	Completado
Autoinforme	19/12/2022	16/01/2023	Completado
Aprendizaje	**12/10/2022**	**16/01/2023**	
Repaso de TypeScript	12/10/2022	16/01/2023	Completado
Aprendizaje de Nest	12/10/2022	16/01/2023	Completado
Aprendizaje de Postgres	12/10/2022	09/11/2022	Completado
Repaso de Angular	10/11/2022	16/01/2023	Completado
Diseño	**12/10/2022**		
Arquitectura de la aplicación	12/10/2022	09/11/2022	Completado
Diseño de diagramas	12/10/2022	09/11/2022	Completado
Casos de uso	12/10/2022	09/11/2022	Completado
Estudio de usabilidad	12/10/2022	09/11/2022	Completado
Prototipos	10/11/2022	18/12/2022	Completado
Pruebas de usabilidad	10/11/2022	18/12/2022	Retrasado: 16/01/2023
Diseño de la identidad gráfica	12/10/2022	09/11/2022	Completado
Desarrollo	**06/03/18**	**02/06/18**	
Configuración del entorno de trabajo	12/10/2022	28/10/2022	Completado
Instalación de Node, Angular y Nest	12/10/2022	28/10/2022	Completado
Desarrollo de la API	12/10/2022	18/12/2022	Completado
Desarrollo del front end	10/11/2022	18/12/2022	Completado
Fase de pruebas	12/12/2022	18/12/2022	Completado
Detección de bugs	12/12/2022	18/12/2022	Completado
Despliegue para producción	19/12/2022	23/12/2022	Completado
Pruebas con usuarios reales	26/12/2022	30/12/2022	Completado
Puesta en marcha en servidor	09/01/2023	13/01/2023	Completado
Carga pública en GitHub	09/01/2023	13/01/2023	Completado

1.1. Tabla de etapas del proyecto

1.6. Breve sumario de productos obtenidos

- Memoria final
- API: Proyecto en Nest
- Frontend: Proyecto en Angular
- Presentación del proyecto

- Documentos complementarios:
  - Endpoints (postman)
  - Funcionamiento MVC (draw.io)
  - Diagrama de flujos (draw.io)
  - Diagrama de entidades (draw.io)
  - Logo ADF (affinity designer)
  - User (affinity designer)
  - Agency (affinity designer)
  - Wireframes (balsamiq)

1.7. Breve descripción de los otros capítulos de la memoria

A continuación se describe brevemente el contenido de los diferentes capítulos de esta memoria y su relación con el proyecto.

1. Introducción

En este capítulo se desarrolla un acercamiento al contexto legal, nacional e internacional, que lleva a plantear el problema planteado para desarrollar este trabajo. Así mismo se especifican los objetivos, impactos en la agenda 2030, el enfoque planteado, la planificación y seguimiento y los productos obtenidos.

2. Administración Digital Fácil

Se plantea el mercado en el que se implantará nuestra solución así como la situación actual del público objeto y la viabilidad de llevarlo a cabo con todos los problemas que pudieran surgir a lo largo del desarrollo.

3. Arquitectura de la aplicación

A lo largo de este capítulo encontraremos un acercamiento al patrón MVC sobre el que se articulará el desarrollo del proyecto así como los diferentes diagramas de flujo, definición de entidades de la base de datos y los posibles casos de uso en función de los roles de usuario.

4. Diseño de la usabilidad

Un acercamiento a las diferentes opciones de usabilidad y la que se considera más indicada para el caso a desarrollar. Se completa todo ello con el diseño de los prototipos de la web a implementar, la identidad gráfica que regirá todo el proyecto y el resultado de las pruebas de usabilidad y uso global de la aplicación.

5. Desarrollo de la aplicación

Este capítulo aborda el desarrollo global de la aplicación a lo largo de sus diversas fases, explicando con más detalle el apartado correspondiente a Nest ya que es la tecnología que se perseguía aprender. En el mismo se pueden encontrar las instrucciones para comprobar algunos de los productos aportados en modo local.

6. Resultados

Respuestas a una batería de preguntas relacionadas con el desarrollo del proyecto, su evolución, éxito y puntos débiles, así como proyección futura del mismo.

7. Glosario

Definición de algunos términos técnicos empleados a lo largo de esta memoria y que pueden no ser comprendidos en su totalidad dentro del contexto.

8. Bibliografía

Mención a las fuentes de información consultadas en el desarrollo de este trabajo.

9. Anexos

Aquella documentación complementaria como encuesta, herramientas de desarrollo e instrucciones de uso.

2. Administración Digital Fácil

2.1. Análisis de mercado

Antes de comenzar a desarrollar el proyecto es interesante hacer un estudio de mercado, para lo que se ha optado por una doble vía de búsqueda. Por un lado, se han analizado las herramientas disponibles similares a la propuesta de este trabajo. Por el otro, se ha analizado el grado de satisfacción de los usuarios con las opciones existentes.

Por último, finalizaremos la sección con un análisis del público objetivo de nuestra solución propuesta, para así, poder continuar con una mejor visión a la hora de realizar el diseño de la aplicación y el estudio de usabilidad.

a. Herramientas actuales en el mercado

Dado que hablamos de una implementación de normativa a nivel estatal por parte de las administraciones públicas, cabría esperar de la implantación a través de algún marco común para todas ellas. Sin embargo, hasta la fecha, ningún organismo competente ha tomado una iniciativa similar, dejando a criterio de los diferentes estamentos su implantación, siendo el más común el de las sedes electrónicas.

Sede electrónica

Según el Ministerio de Educación[1], la sede electrónica es aquella dirección electrónica disponible a través de redes de telecomunicaciones cuya titularidad, gestión y administración corresponde a una Administración Pública, órgano o entidad administrativa en el ejercicio de sus competencias. Es decir, es el lugar en el que se realizarán los diferentes trámites electrónicos de esa entidad concreta, y en la que, se podrá disponer del catálogo correspondiente.

Esto presenta un problema que podemos observar rápidamente echando un simple vistazo a diferentes sedes electrónicas:

- Ministerio de Educación: < https://sede.educacion.gob.es/portada.html >
- Agencia de Protección de Datos: < https://sedeagpd.gob.es/sede-electronica-web/ >
- Generalitat de Catalunya: < https://web.gencat.cat/es/seu-electronica/ >
- Diputación de Burgos: < https://www.burgos.es/sede-electronica >
- Ayuntamiento de Madrid: < https://sede.madrid.es/portal/site/tramites >

Cada una de ellas cuenta con una estructura diferente, una forma de búsqueda distinta. Al ser una gestión individual, la interpretación de espacios sencillos y

[1] < https://sede.educacion.gob.es/que-es.html>

accesibles queda al criterio de cada uno de estos organismos. Y en un entorno como el digital, en el que las capacidades de los usuarios a los que se enfoca son muy diferentes, puede suponer la barrera definitiva entre hacer uso o no de estos medios digitales.

PCOS: Proyectos de acompañamiento digital para la inclusión social

El único punto de información de trámites que podríamos considerar interesante como competencia, de todo lo buscado, es el proyecto PCOS[2] del Ayuntamiento de León.

Consiste en una web en la que se explican los trámites más interesantes para el usuario medio, sin importar el organismo al que corresponda. Lo que encaja con nuestra propuesta. Pero cuenta con un problema importante: es dependiente de un fondo de financiación de la Junta de Castilla y León y los Fondos Sociales Europeos, y como resultado, al concluir esa subvención, el proyecto se convierte en un servicio discontinuado de actualización por parte del Ayuntamiento, en parte porque se desarrollo como parte de un curso de aprendizaje de Bootstrap y muchos de los trabajadores actuales, no tienen conocimiento técnico para continuarlo así.

b. Satisfacción de los usuarios con la relación actual con la Administración (en desarrollo: falta actualizar los datos de la encuesta en este informe y analizar sus resultados)

A lo largo de dos meses se ha realizado una encuesta a través de internet abierta a cualquier ciudadano para que pueda expresar su opinión acerca de la relación de internet.

Tanto las preguntas como las respuestas se pueden consultar en el Anexo 9.1 Investigación sobre el panorama actual de esta memoria.

Analizando los resultados arrojados para una muestra breve de población, con unas características de vida media similares, podemos observar algunos parámetros interesantes.

Cabe destacar que el mayor número de participación de la encuesta se da en las franjas correspondientes a las llamadas Generación X e Y (millennials). Es decir, son una parte relevante de la población activa y son nativos digitales o migrantes digitales con una fuerte implantación de las nuevas tecnologías a nivel formativo y laboral. Además cabe mencionar que las personas que tienen capacidad y herramientas para contestar a una encuesta de estas características cuentan con unas habilidades que les aleja del perfil de analfabetos digitales y por tanto, salvan esa brecha digital en la que muchas veces se respaldan las administraciones.

[2] < https://pcos.puertasdigitales.es/ >

Se puede concretar dentro del perfil que cuentan mayoritariamente con estudios superiores y residen en ciudades donde el acceso a internet no es un problema de infraestructuras, prueba de ello es que más del 96% cuenta con acceso en su domicilio y la totalidad cuenta con dispositivos suficientes para hacerlo. Además todos ellos tienen presencia en alguna red social.

Poco más de la mitad, cuenta con certificado electrónico, lo que nos lleva a deducir que su relación con la tecnología y diversos trámites se ha llevado a cabo en algún momento de su vida.

Con estas características de perfil evaluado, podríamos pensar que no tendrán grandes problemas para la tramitación digital y que, los procedimientos administrativos que necesiten hacer para simplificar su vida cotidiana, serán fáciles y accesibles para ellos. Pero la realidad dista de esta primera conclusión.

La sanidad ha mostrado una gran transformación, fruto entre otras cuestiones, por la reciente pandemia de la covid-19. Lo que ha forzado a usuarios y administraciones a un entendimiento digital que se traduce en que tan solo un 14% sigue prefiriendo comprobar sus datos médicos de manera presencial o telefónica.

Algo similar sucede cuando se pregunta por trámites fáciles y habituales como pedir cita para renovar la documentación o solicitar una vida laboral. Aunque entre un 4 y un 14% de los encuestados, no conocían esa posibilidad o necesitan un medio ajeno.

Pero trámites menos habituales como consultar datos en tráfico, muestran algunos de sus procedimientos como desconocidos o no accesibles para más de la mitad de la población. Y, llama la atención que solo el 27% tenga conocimiento del sistema Tabi para hacerse socio de una biblioteca pública.

En el apartado de experiencia reclaman sencillez en las herramientas. Donde más del 30% necesita otra vía de comunicación o incluso prefiere dejar de hacer cosas si el trámite es digital.

Respuestas como “las herramientas suelen ser engorrosas”, “siempre es difícil o frustrante”, “no son para gente mayor o con poco conocimiento de internet” o “se lo tengo que pedir a mi hijo”, ayudan a comprender que casi el 40% opine que las herramientas de la administración no son para todos y que el 83% confíe en la posibilidad de que una web que aúne los procedimientos más comunes, independientemente de la administración, y bien explicados, sería una opción más fácil de manejar que las actuales.

Podemos concluir, que, aun teniendo un conocimiento a nivel usuario moderado-alto, la propuesta de solución de este trabajo final, podría contar con una buena acogida por parte de los usuarios finales.

c. Público objetivo de nuestra aplicación

Podríamos hablar de dos usuarios claramente definidos: quienes consuman los datos y quienes los faciliten.

En el primer término nos encontraríamos con un público objetivo generalista, en una franja de edad que podría alcanzar hasta los 76 años y desde un inicio joven. Entre los que encontramos usuarios nativos y no nativos digitales, que en ocasiones pueden no haber tenido una amplia formación digital ni un uso reiterado del día a día. Para afrontar a este público es necesario un diseño de usabilidad sencillo y lo más intuitivo posible, abogando siempre por un entorno limpio y libre de grandes artificios.

En el segundo lugar nos vamos a encontrar con un público más específico para grabar los datos, y que serán los verdaderos usuarios de la aplicación desde la parte más lógica. Independientemente de su perfil, tenemos que contar con que nos enfrentamos a usuarios que forman parte de la administración pública y por tanto conocedores, en un mínimo exigible, de herramientas informáticas que les posibilite la grabación de los datos a consumir. Debido a que se tratará de una herramientas profesional para estos usuarios debe buscar ser limpia y lo más eficiente posible, para que no necesiten de numerosos pasos para alcanzar la tarea a realizar con éxito.

2.2. Viabilidad de proyecto

Son varios las factores a considerar a la hora de afrontar un estudio de viabilidad de un proyecto. Desde la viabilidad técnica, de mercado, económica, legal, de conocimientos… Hagamos un breve análisis de las mismas

a. Viabilidad del mercado

Tal y como hemos visto en el estudio anterior del mercado, un producto que aúne la información de los trámites administrativos más habituales tiene cabida dentro de la demanda. Por lo que podemos concluir que hay una **alta posibilidad de viabilidad** en este terreno.

b. Viabilidad económica

Estamos frente a un proyecto que no cuenta con dotación presupuestaria para llevarse a cabo. El mayor gasto es el consumo de horas por parte de la persona que desarrolla el proyecto, que al formar parte de un trabajo final de máster no requiere de remuneración económica alguna. El mayor riesgo en este respecto es la compatibilidad del desarrollo del proyecto con una vida laboral y familiar paralela que pudiese restar horas de dedicación necesarias.

c. Viabilidad legal

Todas las herramientas necesarias para el desarrollo de este proyecto, así como la información necesaria para ser consumida en la API y la documentación requerida para formarse, cumplen con la legislación vigente y ni el desarrollo de la aplicación ni un hipotético uso futuro conllevarían ningún incumplimiento normativo. Siendo **plenamente viable**.

d. Viabilidad técnica

En este apartado encontramos dos subsecciones. Por un lado podemos hablar de la habilidad técnica en la gestión de proyecto, dado que se trata de un reto al que nos enfrentamos por primera vez no se tiene certeza alguna de que los tiempos y márgenes de la planificación y la correspondiente desviación sean los adecuados.

Por otro lado, el apartado técnico a nivel de dispositivos puede suponer un riesgo elevado debido a que se parte con un hardware algo deficiente y se espera su reemplazo a lo largo de este proyecto, lo que podría suponer problemas a la hora de encontrar fallos tanto en la parte física como en la lógica.

Si bien este apartado es **uno de los que mayor riesgo tienen en la viabilidad final** de los objetivos, se cuenta con unos márgenes holgados para poder llevar a cabo la aplicación en las formas propuestas.

e. Viabilidad de conocimiento

A lo largo del proyecto se ha previsto el aprendizaje de nuevas tecnologías nunca antes exploradas como Nest, PostgreSQL y herramientas como Docker, Git o TablePlus, que podrán llevar a retrasos en el desarrollo o problemas de solución de errores que podrían ser más fácilmente resolubles con entornos menos arriesgados.

Por otro lado, el hecho de provenir de un campo profesional muy distante del tecnológico y contar con la única experiencia de la actual formación en la UOC del resto de herramientas y tecnologías, también dan un margen pequeño de error en este punto.

Para tratar de tener el menor riesgo posible en este apartado de viabilidad, que junto con el técnico, son los que podrían arruinar el proyecto, se ha recurrido a formación complementaria y documentación técnica de todos ellos. Además de aprovechar las semanas previas al inicio para repasar lo aprendido a lo largo del máster y que puede ser de interés para el normal desarrollo actual.

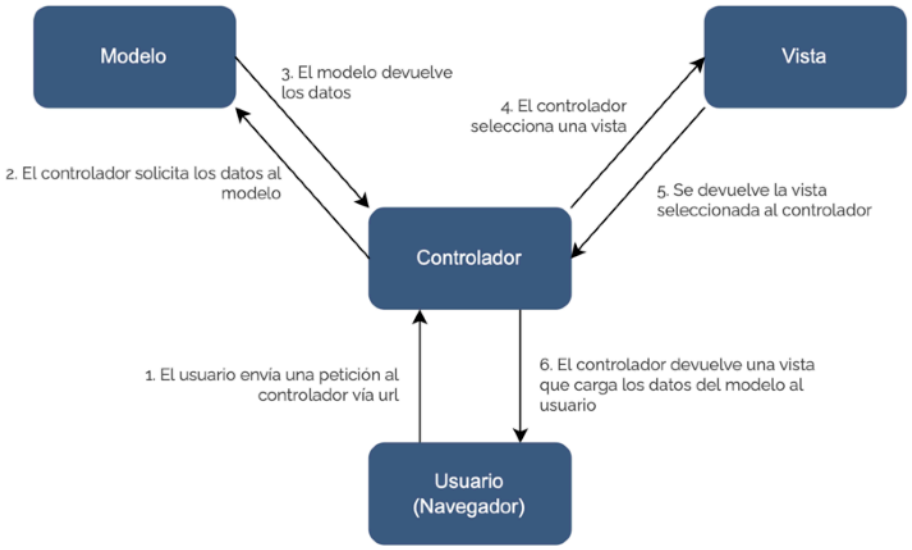
3. Arquitectura de la aplicación

3.1. Patrón planteado

Se plantea la aplicación siguiendo un patrón MVC, un modelo de arquitectura que separa el código dependiendo de sus responsabilidades, de manera que mantiene diversas capas que se encargan de tareas muy concretas, lo que ofrece beneficios muy interesante. Las siglas representan las capas Modelo, Vista y Controlador.

En la primera capa se trabajará con los datos y la base de datos, y se verá reforzada por el uso de la librería TypeORM para trabajar con los datos de manera de forma abstracta.

La capa de la vista se ocupa de mostrar la salida, es decir, crear la interfaz del usuario. Mientras que la capa del controlador será la encargada de responder a las acciones entre ambas. Tal y como podemos observar en la siguiente imagen.



3.1 Funcionamiento de una aplicación MVC

Mantener esta estructura de manera modular permite un desarrollo mucho más fácil de entender, mantener y expandir.

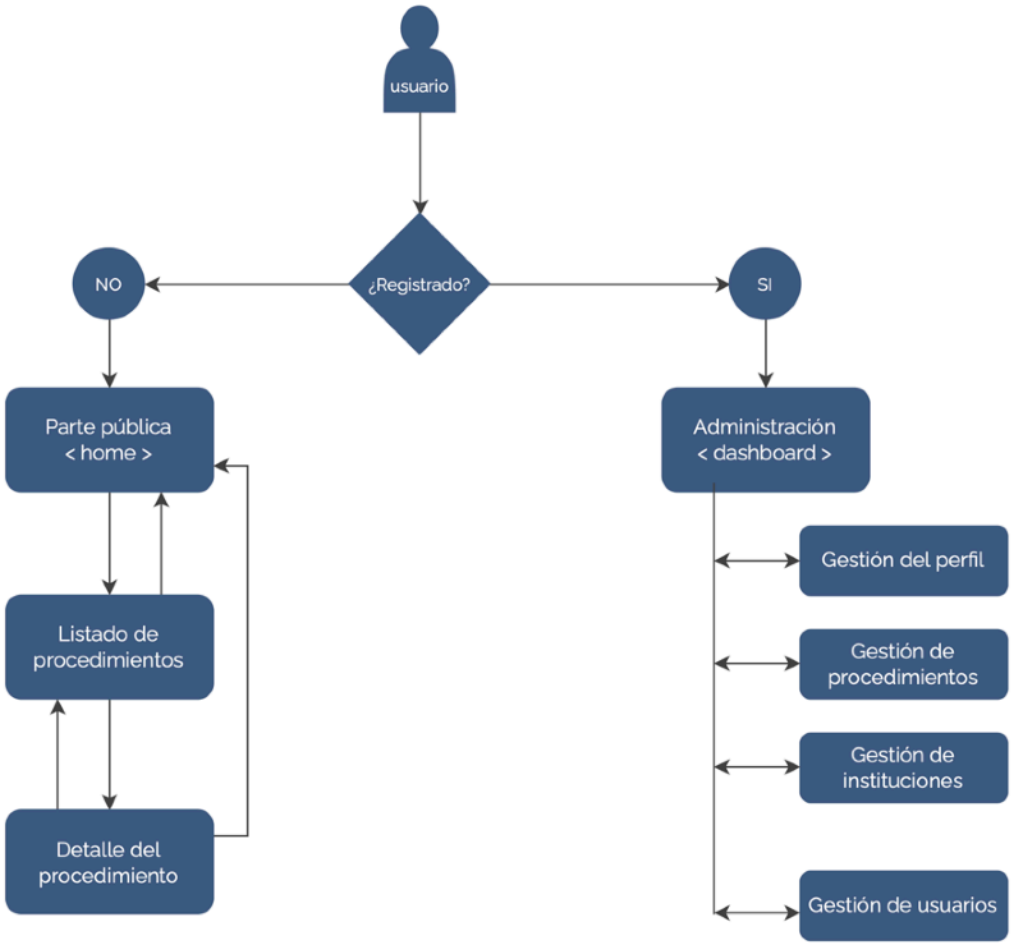
El lenguaje de programación sobre el que recaerá la mayor parte del peso del desarrollo será TypeScript ya que es el empleado en los dos principales frameworks utilizados para responder a las necesidades del backend con la creación de la API mediante Nest, y las necesidades del frontend mediante Angular. Ambos frameworks cuentan con un paradigma muy parecido al modelo propuesto. Mientras que los datos serán manejados mediante PostgreSQL en una gestión con la API REST.

La interfaz final del usuario se basará en el modelo SPA (single page application) que genera una única página que no se recarga durante su uso, lo que ayuda a una mayor respuesta que una web tradicional.

3.2. Diagrama de flujo

La aplicación que queremos desarrollar cuenta con dos escenarios, por un lado la parte pública a la que tiene acceso cualquier usuario con conexión a internet y que no necesita registro para su uso.

Por otro lado tenemos al usuario que suministra información que dependiendo de su rol , tal y como veremos en los casos de uso, tendrá más o menos acceso a la parte del backoffice.



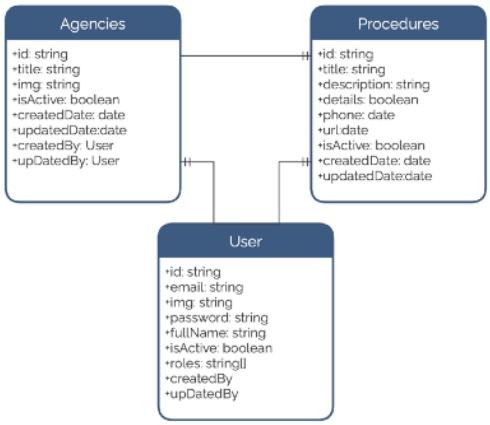
3.2 Diagrama de flujo de Administración Digital Fácil

Podemos observar en el diagrama de flujo anterior cuál será el recorrido de ambos escenarios. Por un lado el usuario no registrado podrá ir de la página principal a los listados de procedimientos de las diferentes agencias y volver. Podrá ir del listado al detalle del procedimiento y volver, o regresar a la página inicial.

Mientras que un usuario registrado, dependiendo de su rol, podrá moverse de una manera menos lineal a lo largo de casi todas las herramientas a las que tenga acceso. Este apartado será desarrollado en mayor profundidad en el apartado de Casos de uso.

3.3. Diagrama de entidades

Para desarrollar la aplicación necesitamos definir cuáles serán las entidades que serán representadas en la base de datos y su relación entre ellas. En el siguiente diagrama podemos tener un primer acercamiento a las mismas:



3.3 Diagrama de entidades de la BD

Como podemos apreciar necesitaremos gestionar tres elementos diferentes y relacionados entre ellos. Por un lado necesitamos definir a los usuarios por una doble razón, es necesario una gestión que permita el acceso a la aplicación y por otro lado nos servirá para tener una relación referencial de quién crea y actualiza las instancias de las otras dos entidades.

El caso de los procedimientos también es fácilmente asumible que necesitará de su propio espacio, ya que, al fin y al cabo, mostrar la información contenida en estos es el fin último de la aplicación.

En último lugar tenemos la entidad relativa a las instituciones, de esta manera podemos dar un sentido de ordenación a los procedimientos en función de su categoría.

Las relaciones que se establecen entre sí son obligatorias, es decir, no se puede crear una institución sin referencia a su usuario creador, ni un procedimiento sin una relación con la institución a la que pertenece:

- Relación Usuario - Institución (1:N): Un usuario puede crear tantas instituciones como considere oportuno, mientras que una institución solo tendrá un creador.

- Relación Usuario - Institución (1:N): Un usuario puede actualizar tantas instituciones como considere necesario, mientras que una institución solo conservará la relación del último "actualizador". Esta relación podría entenderse también como N:N, es decir, que la institución pudiese almacenar tantos

usuarios como actualizaciones realizase, pero para el caso que ocupa a la aplicación no es relevante tanto detalle.

- Relación Usuario - Procedimiento (1:N): Un usuario puede crear tantos procedimientos como quiera, mientras que un procedimiento solo tendrá un usuario creador.

- Relación Institución - Procedimiento (1:N): Una institución puede contar con tantos procedimientos como se desee, mientras que un procedimiento solo será tramitado en una institución.

3.4. Casos de uso

A continuación se explicará brevemente el camino y uso que puede darse a la aplicación en función de las características de cada usuario.

Aunque la aplicación es responsiva y se adapta tanto a dispositivos móviles como ordenadores u otros dispositivos con mayor pantalla, cabe indicar que todos los roles registrados se contemplan para un perfil profesional que desarrollará su uso en soportes superiores a móviles.

a. Usuario no registrado

El usuario no registrado será el público general. A pesar de contar con un amplio abanico de perfiles tecnológicos posible, la aplicación se desarrolla con la necesidad de que sea lo más simple posible para adaptarse a los usuarios con los conocimientos mínimos de acceso a internet.

El usuario tendrá una página principal en la que verá cada una de las instituciones disponibles y a través de las cuales podrá navegar hacia sus procedimientos determinados que se mostrarán con el mismo formato sencillo y un botón visible para poder acceder a los datos explicativos de cada uno.

El usuario siempre puede volver al inicio mediante el logotipo en el *header*, y mediante el correspondiente botón que indica la vuelta al inicio. Además para la navegación hacia el punto anterior, también existirá un botón volver, asumiendo que en muchos casos los usuarios pueden emplear esa opción del botón de su navegador.

b. Usuario registrado: user

Es el rol más básico con acceso a la gestión de la aplicación. Es un usuario que tendrá unos conocimientos mínimos de acceso informático y que desarrollará su profesión con herramientas similares. Preferiblemente será miembro de la administración y su labor se limitará a rellenar los procedimientos.

Al acceder a la aplicación tendrá una visión de sus datos personales que podrá actualizar mediante el correspondiente botón que le permitirá ver y actualizar datos de su perfil como el nombre, el email y la contraseña, en cambio no podrá modificar ni su rol ni su activación. Aunque en el hipotético caso de estar desactivado el sistema de login no le permitirá el acceso y necesitará contactar con un rol superior.

Puede crear un procedimiento, aunque esté no será público hasta tener validación superior.

Tiene acceso al listado de procedimientos generales desde donde podrá actualizar los que considere necesarios, nunca la visibilidad. Podrá acceder también a un espacio en el que ver solo los procedimientos de su creación y gestionarlos del mismo modo que los generales.

Existe la posibilidad de acceder a la parte pública sin perder la condición interna de registro y poder alternarla en todo momento con la de gestión. De igual modo, existe un menú de usuario desde el que salir de la aplicación.

Los usuarios registrados cuentan además con un menú lateral desde el que acceder a todos los apartados correspondientes sin necesidad de hacer circuito de pasar por diferentes páginas para llegar a otras.

c. Usuario registrado: superuser

Es el siguiente rol de la aplicación, tiene acceso a todos los aspectos que user y además puede validar la visibilidad de los procedimientos, así como eliminarlos.

Se trata de un perfil con conocimientos medios tanto a nivel técnico como administrativo, por lo que antes de publicar los procedimientos tendrá capacidad para verificar su contenido.

Además tendrá habilitada una sección en la que encontrar los procedimientos no activos, es decir los que no se muestran. Y también podrá crear y actualizar instituciones, aunque en este caso no podrá borrarlas ni hacerlas públicas.

d. Usuario registrado: crossuser

Este usuario tiene un nivel superior al superuser, es decir, puede llevar a cabo las mismas acciones que los anteriores y navegar por las secciones con la misma libertad. Además tendrá acceso a los apartados de las instituciones no activas para poder hacerlas públicas.

Si bien el nivel de conocimiento para el manejo de la aplicación no requiere de grandes conocimientos, este tipo de usuario, y el siguiente, tienen que tener al menos un mínimo de cómo funciona la aplicación a la hora de eliminar instancias. Eliminar una institución hará que se eliminen en cascada todos los procedimientos contenidos en ella, por lo que

siempre será la última opción, recomendando siempre la desactivación frente a la eliminación.

Este usuario podrá gestionar a los demás usuarios, así como desactivar las cuentas. Si bien es el primer usuario que tiene acceso a todas las secciones de la aplicación, tendrá limitadas las funciones en la gestión de usuarios, a los que no podrá eliminar ni nombrar administradores. Se entiende que esa necesidad tendrá que transmitírsela a un usuario con rol admin.

e. Usuario registrado: admin

Se trata del último caso de usuario registrado y que como con los anteriores tendrá acceso a todas las páginas de navegación de sus predecesores y a alguna funcionalidad más como crear o nombrar administradores de la aplicación y eliminar usuarios.

Los conocimientos técnicos necesarios, al igual que sucedía con el crossuser, no son necesariamente elevados, pero sí es muy importante tener en cuenta el modo de eliminación de los diferentes estamentos como la ya mencionada eliminación en cascada para los procedimientos si se elimina una institución; o la pérdida referencial del creador o último modificador si se eliminan usuarios. Razón por la que siempre que sea posible, se recomendará no eliminar estas categorías.

4. Diseño de la usabilidad

4.1. Estudio de la usabilidad

Se inicia el acercamiento a la parte del usuario con una premisa clara: **simplicidad**. No podemos presentar una herramienta que busque hacer más fácil el recorrido al usuario si al mismo tiempo complicamos la supuesta solución.

El desarrollo será consumido, dependiendo del perfil, en dos medios muy concretos: escritorio para los que doten de contenido y utilicen la aplicación como usuarios registrados; y cualquier dispositivo, con tendencia a los móviles, por parte de los consumidores de la aplicación o los usuarios no registrados. Por esa razón el planteamiento a la hora de diseñar los prototipos no podrá ser ni desktop-first ni mobile-first, por lo que será mejor optar por la implementación de dos patrones de diseño orientados a la retícula para este fin:

- Tiny tweaks: se ajusta en forma de pequeños cambios como el padding o los márgenes que se pueden reducir en dispositivos móviles, o el tamaño de la fuente.

- Mostly fluid: un patrón basado en la cuadrícula fluida que hará que los elementos se ajusten a lo ancho de la columna en dispositivos grandes o cambien su posición a una forma vertical si el dispositivo se hace más pequeño.

Otra cuestión que se quiere contemplar en la aplicación, son los principios de usabilidad de Nielsen:

- Visibilidad del estado del sistema: intentar explicar al usuario en todo momento de dónde se encuentra y ofrecerle la información necesaria en todo momento.

- Relación entre el sistema y el mundo real: mediante el empleo del lenguaje fácil o lo más sencillo posible para que pueda ser entendido por un usuario sin grandes conocimientos técnicos.

- Control y libertad del usuario: facilidad de acceso siempre al menú inicial para que el usuario pueda moverse sin miedo y tener siempre una vía fácil de salida. Además, de darle opciones de confirmación en caso de hacer modificaciones sin posibilidad de retorno.

- Consistencia y estándares: uso de estándares establecidos dentro del mundo de internet, como que el logo de la aplicación nos lleve siempre al inicio.

- Prevención de errores: se desarrollará con el propósito de evitar el mayor número de errores posibles por parte del usuario. Además, se alertará con mensajes de confirmación todas aquellas acciones que no puedan ser revertidas, como eliminaciones de la base de datos.

- Reconocimiento antes que recuerdo: el uso de iconos fáciles de entender junto con el texto fácil para que el usuario se sienta en todo momento seguro en la aplicación.

- Flexibilidad y eficiencia de uso: intentar simplificar al máximo el recorrido para tomar acciones.

- Estética y diseño minimalista: el sitio permanecerá tan simple como sea posible.

- Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores: siempre que algún formulario requiera de una información no proporcionada, devolverá un mensaje de error claro al usuario para que sepa encontrar y corregir dicho error

- Ayuda y documentación: se implementará una sección de ayuda en la aplicación, además de toda la documentación correspondiente en cada uno de los frameworks empleados en su desarrollo.

4.2. Prototipos

Antes de empezar a maquetar la aplicación se realizaron una serie de prototipos con la aplicación Balsamiq para poder tener una idea uniforme del proyecto visual.

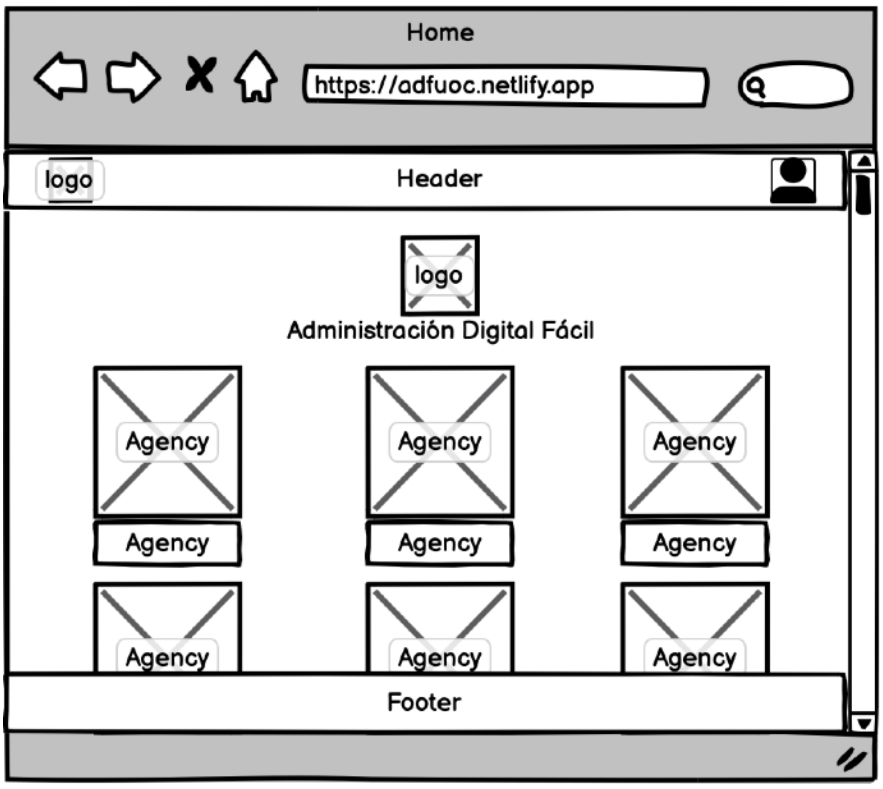
No se desarrollaron todas las pantallas, solo las más relevantes dejando aquellas que por su similitud a otras no era necesario realizar el *wireframe*.

En el caso de la parte pública se consideró necesario hacer un diseño tanto para escritorio como para dispositivos móviles, marcando, como ya se ha mencionado anteriormente, la necesidad de poner énfasis en la responsividad de ese entorno.

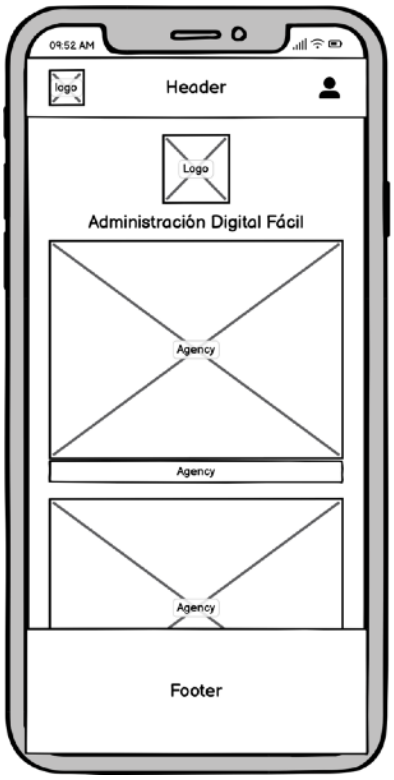
Así podemos ver en las imágenes siguiente el recorrido de un usuario público a través de la página principal en la que encontraremos todas las instituciones en las que sus imágenes en forma de enlaces darán paso al detalle de todos los procedimientos de esa institución.

En el listado de los procedimientos de cada institución podemos encontrarnos con el acceso al detalle de cada uno de los procedimientos.

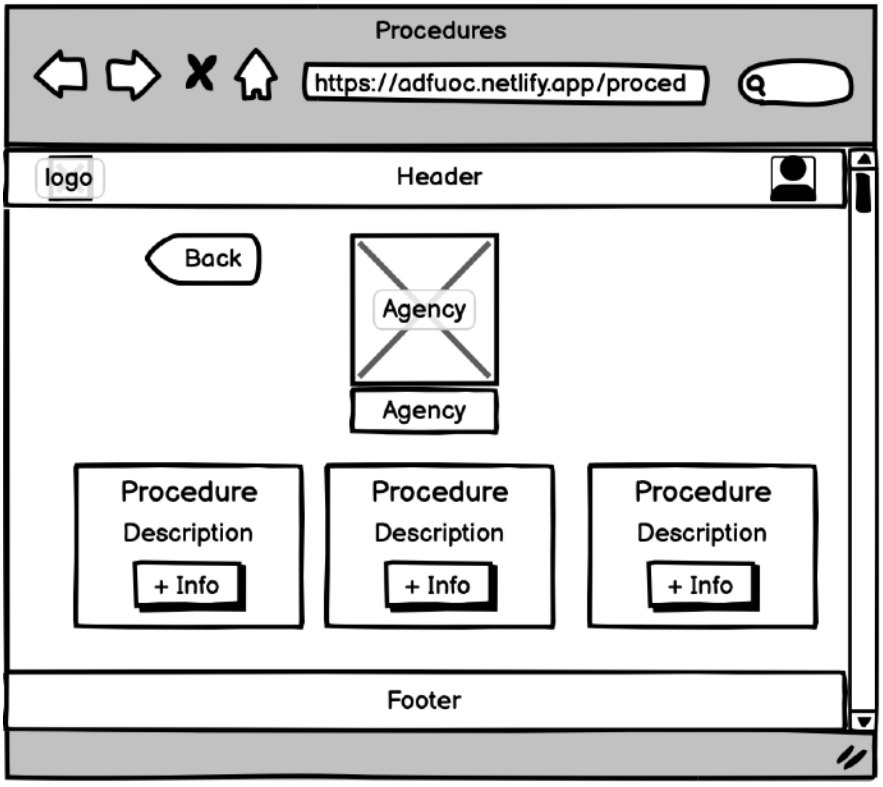
Y en la última pantalla pública veremos el detalle de un procedimiento elegido y como la navegación le permite volver al listado de su institución o a la página de inicio.



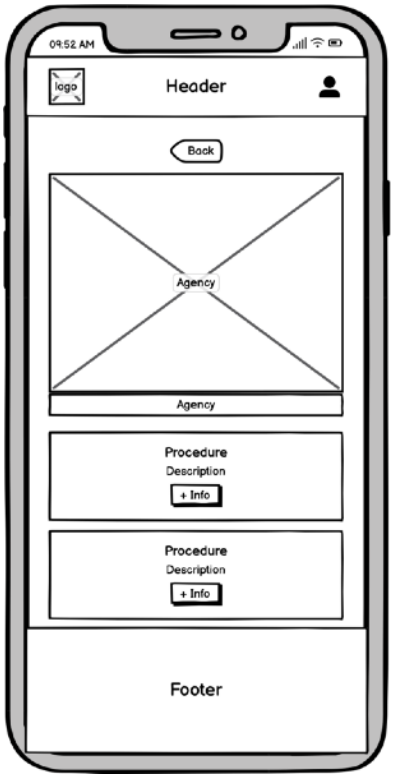
4.1 Wireframe de la página de inicio en escritorio



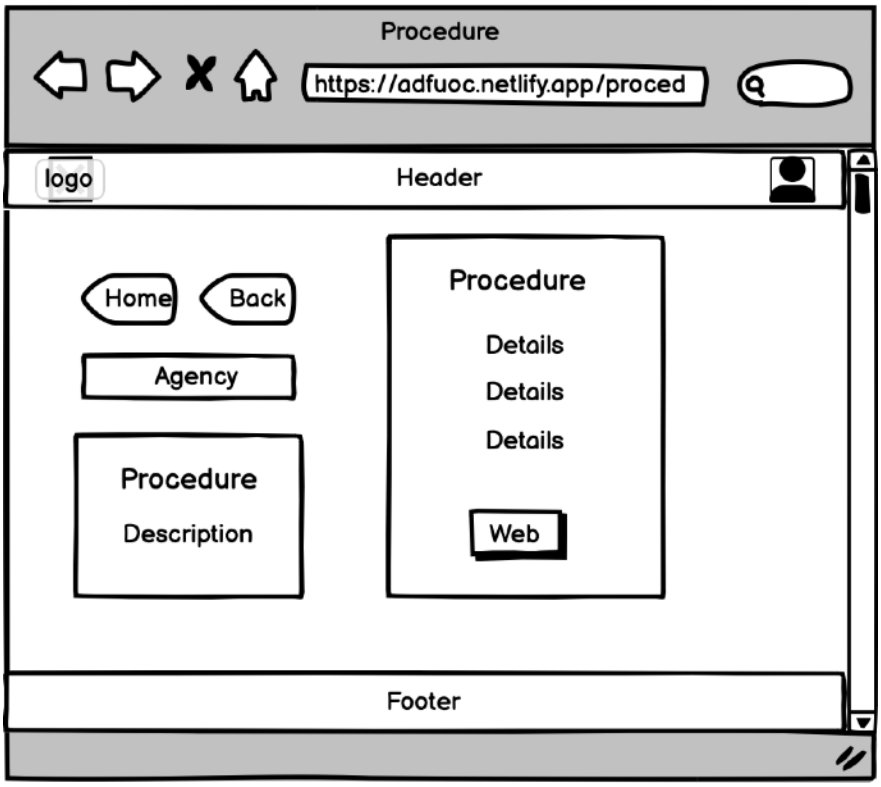
4.2 Wireframe de la página de inicio en móvil



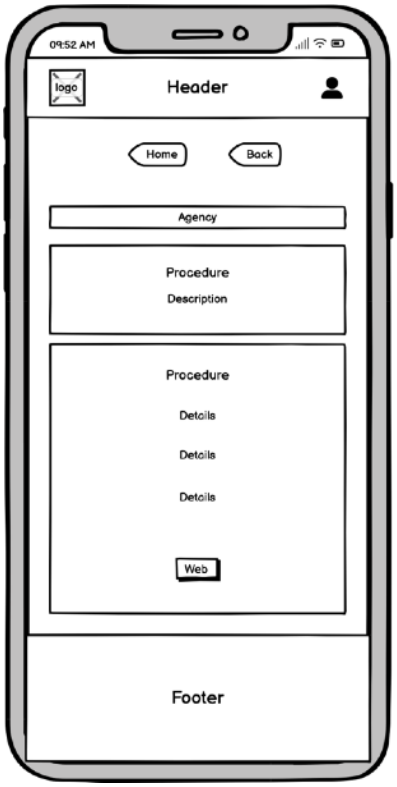
4.3 Wireframe de la página de procedimientos de una agencia en escritorio



4.4 Wireframe de la página de procedimientos de una agencia en móvil

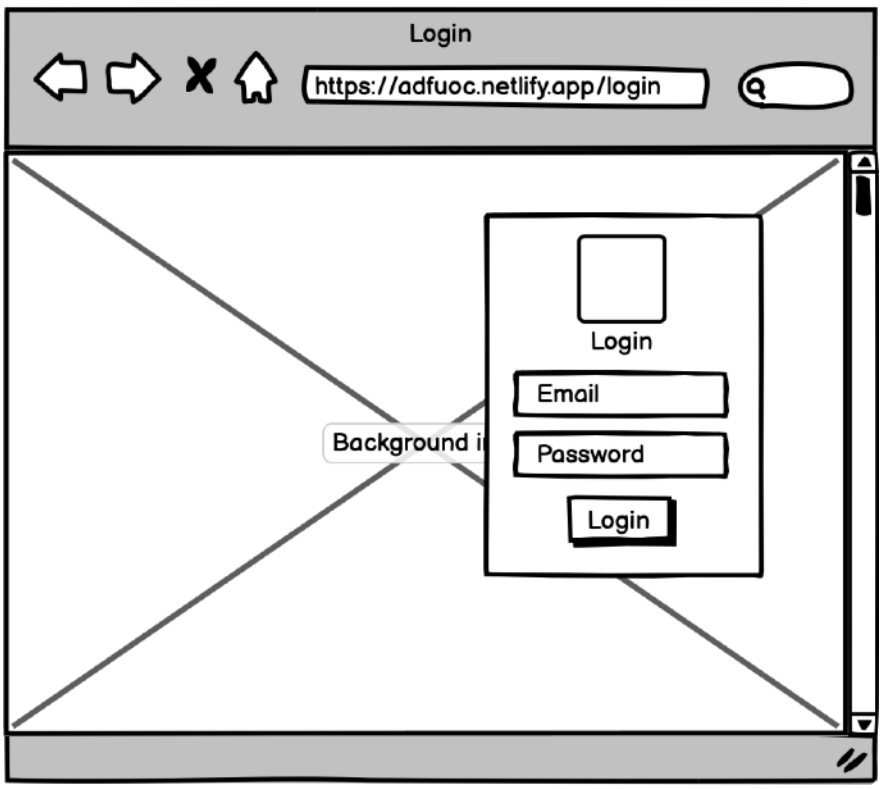


4.5 Wireframe de la página de detalle de un procedimiento en escritorio



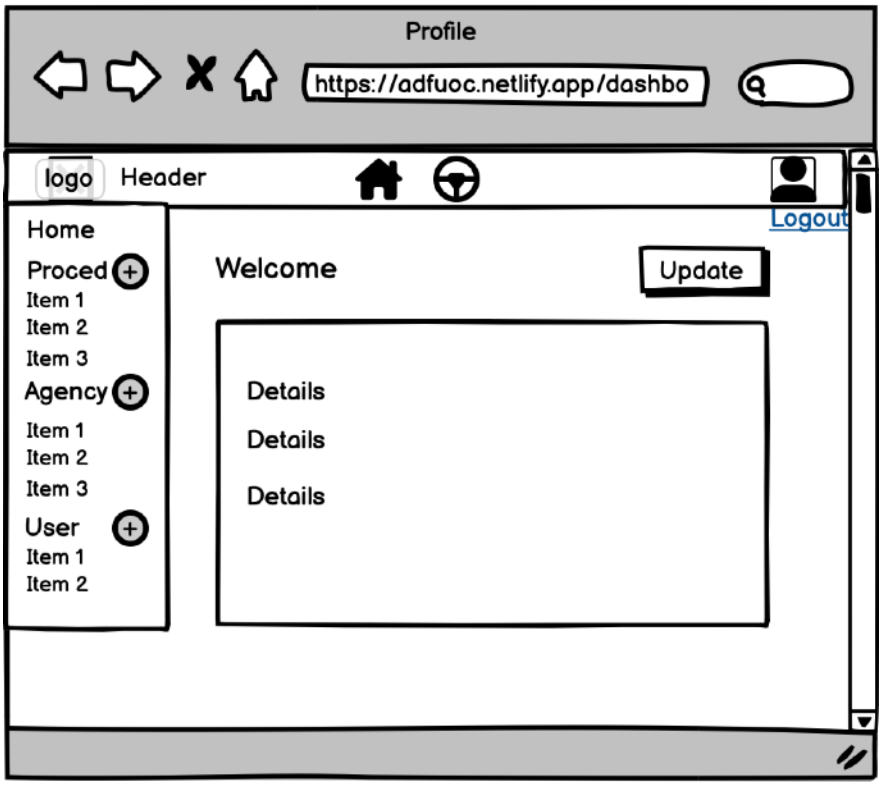
4.6 Wireframe de la página de detalle de un procedimiento en móvil

En cambio, la parte de los usuarios registrados, si bien se han implementado las mismas técnicas para adaptarse a dispositivos móviles, solo se prototipó la versión de escritorio, al ser en ese entorno donde se plantea su utilización.

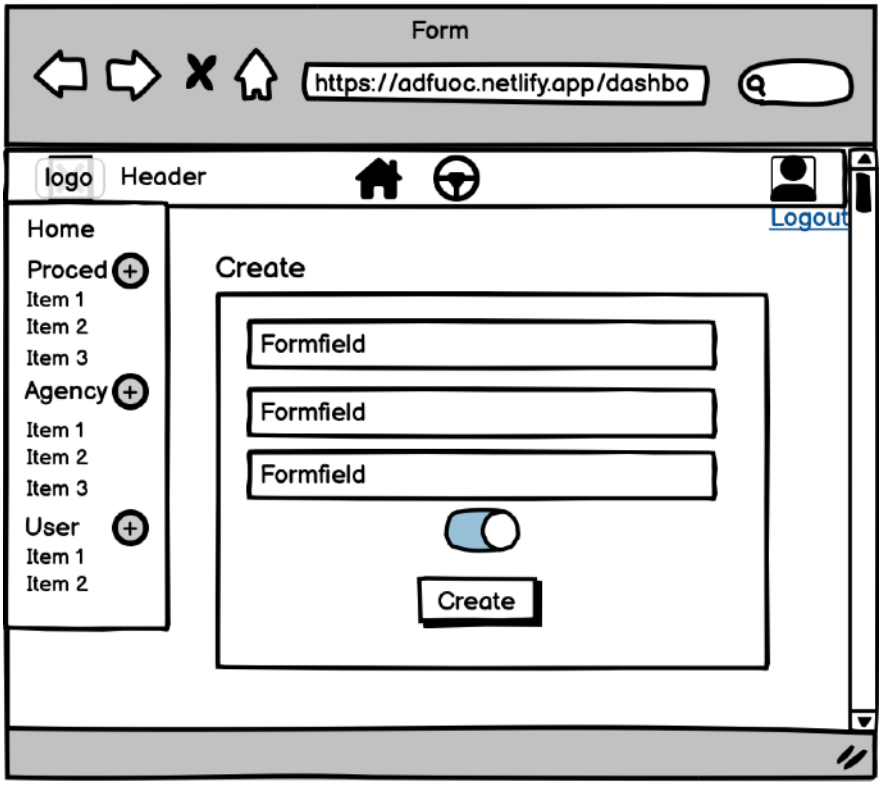


4.7 Wireframe de la página de login

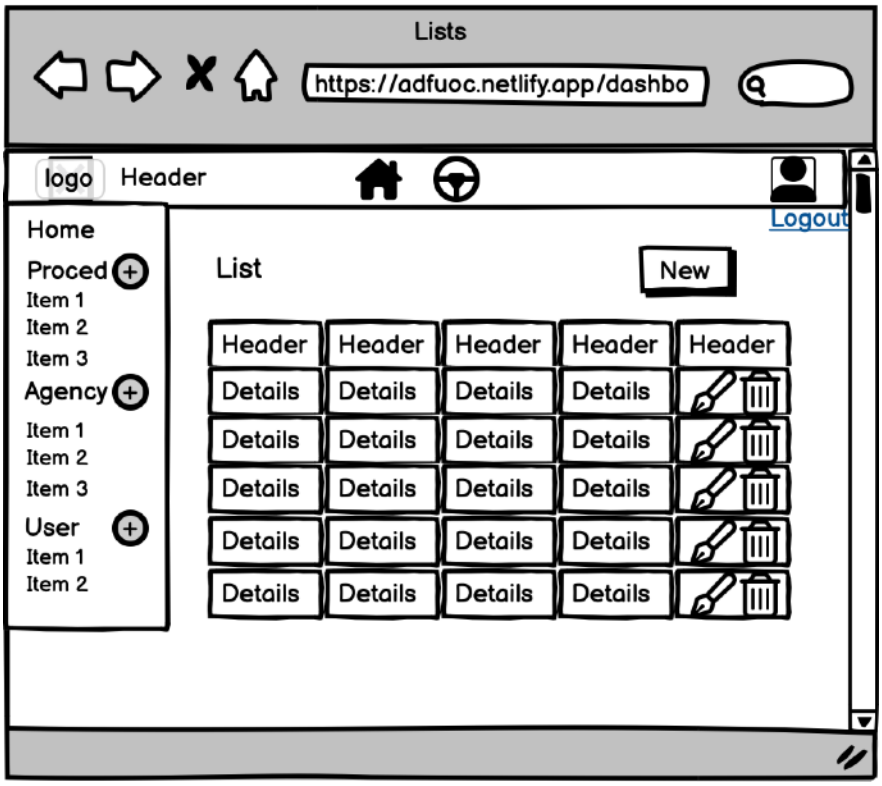
Tampoco se detallaron todas las pantallas, solo aquellas únicas o un modelo de aquellas que podían ser adaptadas a otros planteamientos como listas y formularios.



4.8 Wireframe de la página de inicio del dashboard/detalles del perfil



4.9 Wireframe de una página del backoffice con formularios

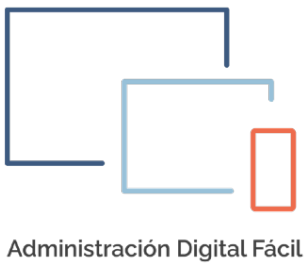


4.10 Wireframe de una página del backoffice con elementos listados

4.3. Identidad gráfica

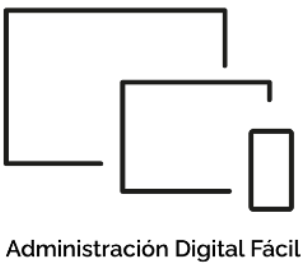
a. Logotipo

El logotipo busca representar el proyecto de un modo sencillo y visual. Para ello juego con formas rectangulares intentando representar diferentes dispositivos de acceso electrónico actuales.



4.11 Logotipo

4.12 Logotipo con fondo



4.13 Logotipo blanco y negro

b. Colores

La paleta cromática elegida para la identidad gráfica del proyecto, y que acompañará no solo al logotipo, también a los colores predominantes en la web y la memoria se componen por tres tonos azulados, propios de muchas administraciones públicas a nivel mundial, y de un resalte anaranjado. Siendo completada por un gris oscuro para los textos, según se refleja en los siguientes parámetros:

C|83 M|58 Y|17 K|10
R|61 G|90 B|128
HEX|#3D5A80

C|38 M|10 Y|3 K|0
R|152 G|193 B|217
HEX|#98C1D9

C|12 M|1 Y|3 K|0
R|224 G|251 B|252
HEX|#E0FBFC

C|0 M|67 Y|71 K|0
R|238 G|108 B|77
HEX|#EE6C4D

C|78 M|72 Y|67 K|27
R|64 G|64 B|64
HEX|#404040

c. Tipografía: Raleway

Todos los textos se llevarán a cabo con la tipografía, perteneciente a la colección de Google Fonts[3]. Se trata de un diseño de Matt McInerney, Pablo Impallari y Rodrigo Fuenzalida perteneciente a la familia sans seria y con numerosas variantes de estilo tanto en el texto regular como en la forma cursiva, lo que ofrece numerosas posibilidades.

A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n ñ o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0

4.4. Pruebas de usabilidad

Debido a un retraso en el proyecto, estas pruebas de usabilidad con personas reales se han llevado en paralelo con las pruebas con usuarios reales en menos tiempo y actores de lo necesario para poder tener un análisis exhaustivo por lo que solo podemos ofrecer una visión general de los resultados.

En cambio, y a pesar de la corta muestra de población y el poco tiempo que se ha podido llevar a cabo podemos concluir que ha sido de gran utilidad, ya que muchas veces es necesaria una voz externa para hacernos ver mejoras que no se nos habría ocurrido en el desarrollo del proyecto sin intervención exterior.

[3] <https://fonts.google.com/specimen/Raleway>

Usuario de la parte pública	
Datos del usuario	
Mujer, 65 años Nivel medio-bajo en competencias digitales Sin certificado digital	Utilizó un ordenador en su vida profesional pero solo conocía las herramientas necesarias para desarrollarlo. Actualmente maneja WhatsApp y es capaz de navegar en internet desde el ordenador y con dificultad en el móvil, pero lo hace poco por miedo.
Resultado de usabilidad	
Satisfactorio pero puede mejorar	- No terminó de comprender que podía volver al inicio con el icono del header - Se liaba un poco cuando había que desplazarse verticalmente - Propuso un buscador para "poder poner lo que buscaba"
Resultado global	
Satisfactorio	Al principio le costaba entender el concepto y se ponía nerviosa, en cuanto fue capaz de encontrar un par de trámites que se le pidieron, empezó a explorar con gran facilidad por el resto de la aplicación.

4.14. Tabla de evaluación de un usuario en la parte pública

Usuario de la parte privada	
Datos del usuario	
Mujer, 45 años Nivel medio en competencias digitales Con certificado digital	Utiliza un ordenador en su vida profesional, tramita expedientes con él y realiza trámites con certificado electrónico sin dificultad. Su formación académica es del área social pero entiende la tecnología como un aporte a su desarrollo profesional. En su vida personal maneja sin dificultad dispositivos móviles, tiene ordenador portátil y ayuda a personas de su entorno con trámites electrónico.
Resultado de usabilidad	
Satisfactorio con cierto grado de mejora	- No usó ninguno de los botones del header hasta que se le propuso, le parecían poco claros e innecesarios - Originariamente no sabía si el botón + servía para añadir elementos o para abrir un desplegable - Propuso un buscador, la posibilidad de ordenar las tablas y agrupar por alguna característica específica
Resultado global	
Muy satisfactorio	Solo hubo que preguntar por el no uso de los botones del header. Fue capaz de realizar satisfactoriamente todos los procesos de la aplicación, previa explicación del funcionamiento de eliminación en el borrado de instituciones y usuarios.

4.15. Tabla de evaluación de un usuario en la parte privada

5. Desarrollo de la aplicación

5.1. Desarrollo del Back-end

Para el desarrollo del backend se utiliza el framework de Nest y la gestión de base de datos mediante PostgreSQL, todo ello se ha desarrollado con la ayuda de Docker y TablePlus en lugar de la instalación completa en el equipo y PgAdmin.

a. Herramientas necesarias para el backend durante la fase de desarrollo

Para llevar a cabo este backend se ha hecho uso del gestor de paquetes npm, aunque si bien es cierto que podría haberse desarrollado con otros similares como yarn, sin grandes cambios más allá de algunas llamadas diferentes a la hora de la instalar los paquetes necesarios y/o recomendados por la documentación oficial de Nest.

La base de datos se asienta sobre PostgreSQL y se gestiona mediante la librería TypeORM, que ayuda a crear y manipular la base de datos sin tener que trabajar directamente con SQL y simplifica el mantenimiento de los datos.

El proyecto se ha completado con dos herramientas más: Docker y TablePlus.

La propia página de Docker[4] lo define como "una plataforma diseñada para ayudar a los desarrolladores a crear, compartir y ejecutar aplicaciones modernas", es decir, estamos frente a una especie de "virtualización ligera" con la que podemos empaquetar entornos que posteriormente desplegaremos en cualquier sistema que disponga de esa tecnología; y se diferencia de las máquinas virtuales en que las segundas necesitan contener todo el sistema operativo a "virtualizar" mientras Docker aprovecha el sistema sobre el que se ejecuta.

Uno de los mayores beneficios que aporta el uso de Docker a la hora de trabajar con la base de datos es el ahorro de instalación de los diferentes entornos que queramos ejecutar, es más, podemos elegir exactamente qué versión del sistema de gestión de base de datos que queremos y podemos trabajar en una más avanzada o menos actualizada en otro proyecto desarrollado en paralelo en el mismo equipo sin necesidad de migraciones de unos u otros gracias a que cada contenedor de Docker es completamente independiente.

Para trabajar con bases de datos necesitamos una herramienta de gestión. Postgres tiene como entorno más habitual pgadmin, sin embargo en esta ocasión y por un motivo similar a Docker se ha optado por utilizar TablePlus[5], un entorno para el manejo de bases de datos que nos posibilita manejar diferentes tipos de bases tanto SQL como NoSQL sin

[4] < https://www.docker.com/>

[5] < https://tableplus.com/>

necesidad de tener que instalar una diferente para cada proyecto. Algo que se ha utilizado bastante ya que en el aprendizaje de Nest para este proyecto se ha probado a hacer pruebas tanto con Postgres como con MongoDB.

b. Cómo utilizar la API en desarrollo

Tal y como puede consultarse en el archivo *README.md* del proyecto, son necesarios unos pocos pasos para poder levantar el proyecto en desarrollo para hacer las comprobaciones necesarias.

En primer lugar tenemos que ubicar el archivo del backend en la carpeta que decidamos utilizar en nuestro equipo de manera local, o clonarlo si decidiésemos utilizar el repositorio de Github[6].

Con ayuda de la consola haremos la instalación del proyecto y todas sus dependencias, propias y de terceros. Si utilizamos npm como gestor de paquetes bastará con emplear la instrucción *npm install* o su forma corta *npm i*.

Si optáramos por conseguir este proyecto mediante el repositorio de Github será necesario clonar el archivo .env.template a .env y actualizar las contraseñas de la base de datos para que coincidan con la que hemos configurado, en el caso actual en TablePlus. Además de ser el lugar en el que tendremos acceso a nuestra clave de encriptación para los JWT (JSON Web Token) con los que encriptaremos a los usuarios autenticados de la aplicación.

El siguiente paso será levantar la base de datos con la ayuda de Docker mediante la instrucción *docker-compose up -d*.

Pondremos en marcha el proyecto en modo de desarrollo gracias a la instrucción *npm run start:dev*.

A partir de aquí tendremos acceso a los distintos endpoints a través de Postman, el navegador en la ruta que indique el puerto de ejecución, por defecto el 3000, o a través de las peticiones que podamos realizar desde el cliente del frontend.

Si queremos rellenar rápidamente una colección de datos para poder hacer pruebas, se recomienda ejecutar en primer lugar el método GET Create Seed, un método que más tarde se desarrollará, que elimina todos los datos existentes en las tablas y carga nuevos datos en todas ellas con las que poder realizar prácticas.

[6] < https://github.com/saraemdev/adf.backend>

c. Paquetes instalados

- Nest CLI: https://www.npmjs.com/package/@nestjs/cli
- Config: https://www.npmjs.com/package/@nestjs/config
- TypeORM: https://www.npmjs.com/package/@nestjs/typeorm
- Postgres: https://www.npmjs.com/package/pg
- Class-validator: https://www.npmjs.com/package/class-validator
- Class-transformer: https://www.npmjs.com/package/class-transformer
- UUID: https://www.npmjs.com/package/uuid
- Bcrypt: https://www.npmjs.com/package/bcrypt
- @nestjs/passport: https://www.npmjs.com/package/@nestjs/passport
- @nestjs/jwt: https://www.npmjs.com/package/@nestjs/jwt
- Passport-jwt: https://www.npmjs.com/package/passport-jwt
- @Nestjs/swagger: https://www.npmjs.com/package/@nestjs/swagger

d. Postman

Postman[7] es una plataforma que nos facilita la creación de APIs ya que podemos ir probando los resultados de la misma sobre respuesta http según vamos desarrollándolas. Nos permite guardar las diferentes llamadas a los endpoints de manera sencilla lo que nos facilita el continuo testing de funcionalidades dependiendo de las llamadas que necesitemos realizar y contempla numerosas posibilidades desde los datos que puede requerir la petición a través de parámetros o body, hasta diferentes maneras de autenticarse o modificar los encabezados.

Para utilizar la plataforma es necesaria una cuenta de usuario y tener el proyecto a comprobar en funcionamiento. Además nos permite exportar nuestra librería de llamadas para poder compartirlas con otros desarrolladores del proyecto tanto del backend como el frontend.

Como parte del proyecto se adjunta a esta memoria la colección de peticiones desarrolladas para la API del mismo bajo el título ADF.postman_collection.json, que puede ser importada para hacer las comprobaciones oportunas.

[7] < https://www.postman.com/ >

Si bien Postman ofrece la posibilidad de crear una documentación en línea a partir de la colección, se ha optado por otra herramienta, recomendada por el equipo de Nest, para llevar la documentación del proyecto.

e. Entidades del proyecto

Como ya se definió en el apartado "3.3. Diagrama de entidades" de esta memoria, la creación de la base de datos se hace entorno a tres tablas y sus relaciones. En esta ocasión no es necesario crear las tablas, ni picar código SQL como en otro tipo de proyectos gracias a la gestión que TypeORM hace por nosotros, pero sí es importante definir los elementos que necesitamos.

Siguiendo la estructura propuesta en el diseño y sus propiedades, el proyecto concluye con las siguientes entidades y relaciones:

- Agency: Encargada de gestionar las instituciones. Relacionada con la entidad Procedures en una relación uno a muchos, y con la entidad User en relaciones muchos a uno.

- User: Encargada de gestionar los usuarios que podrán autenticarse en el sistema. Relacionada con la entidad Agency y Procedure en relaciones de tipo uno a muchos.

- Procedure: Encargada de gestionar los procedimientos. Relacionada con la entidad Agency en una relación del tipo muchos a uno, donde cada institución puede contar con un gran número de procedimientos, y con la entidad User en relaciones muchos a uno.

f. Breve descripción del desarrollo del backend

La gestión más habitual para desarrollar un proyecto con Nest es utilizar las estructuras que, por defecto, nos facilita el framework a través de la línea de comando. Para ello es necesario instalar la interfaz Nest CLI que nos facilita las herramientas necesarias tanto para la inicialización del proyecto como para su desarrollo y mantenimiento. Se puede consultar la referencia a esta instalación en la documentación oficial del framework[8] y en el apartado Paquetes instalados de esta misma sección.

Una vez instalado el framework podemos comprobar su ejecución, en desarrollo, mediante la instrucción npm run start:dev y visualizar su resultado en un navegador en el puerto local 3000, a menos que se decida cambiar por otro a nuestra elección.

La estructura inicial que genera Nest es muy parecida a Angular, y como en el framework de frontend nos encontramos con numerosos archivos de pruebas por defecto. En esta

[8] < https://docs.nestjs.com/cli/overview>

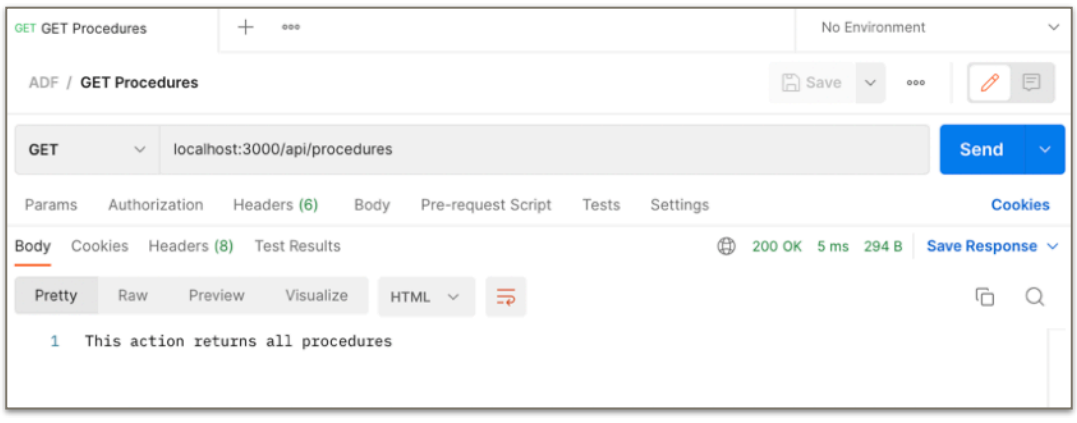
ocasión no vamos a implementar las pruebas unitarias, ya que utilizaremos el entorno de desarrollo de APIs Postman para monitorizar los endpoints y comprobar su funcionamiento. Por esta razón, y para evitar un proyecto demasiado grande con documentos que no se utilicen, se eliminarán los archivos de pruebas y aquellos que no tengan uso específico en el desarrollo como app.service.ts o app.controller.ts, ya que todo ello se manejará desde los componentes desarrollados y, si en el futuro fuesen necesarios, podríamos volver a incorporarlos.

El siguiente paso es la configuración de la base de datos. Podríamos instalar PostgreSQL y gestionarlo desde la herramienta oficial de pgAdmin, pero en esta ocasión, como ya se ha indicado se hará mediante Docker y TablePlus. Para esta gestión creamos el archivo docker-compose.yaml en la raíz del proyecto con los datos necesarios para levantar el contenedor que albergará nuestro proyecto y el archivo .env para configurar las variables de entorno. Mientras en TablePlus generaremos una nueva conexión PostgreSQL con los datos del las variables de entorno.

La conexión de Postgres con Nest utilizaremos la integración de TypeORM, para lo que utilizamos los paquetes recomendados en la documentación oficial de typeorm, que si bien hace el ejemplo con mySQL, sustituimos ese paquete con la correspondiente a postgres. Una vez instalados los paquetes necesarios, configuramos la conexión en el módulo principal de la api.

Con todo configurado comenzamos a crear las entidades necesarias que darán lugar a cada una de las tablas, ya que cada entity será una representación del objeto en la base de datos. En primer término usaremos la CLI (nest g res procedures --no-spec) para que nos agilice el trabajo del CRUD (tenemos que seleccionar la opción REST API) en los recursos que vamos a emplear y dentro de cada uno definimos la entidad correspondiente, así como las relaciones que puedan tener con otras entidades; todo este proceso se simplifica gracias a los decoradores de TypeORM. El último paso para transformar la entidad en tabla será importarla dentro del módulo correspondiente.

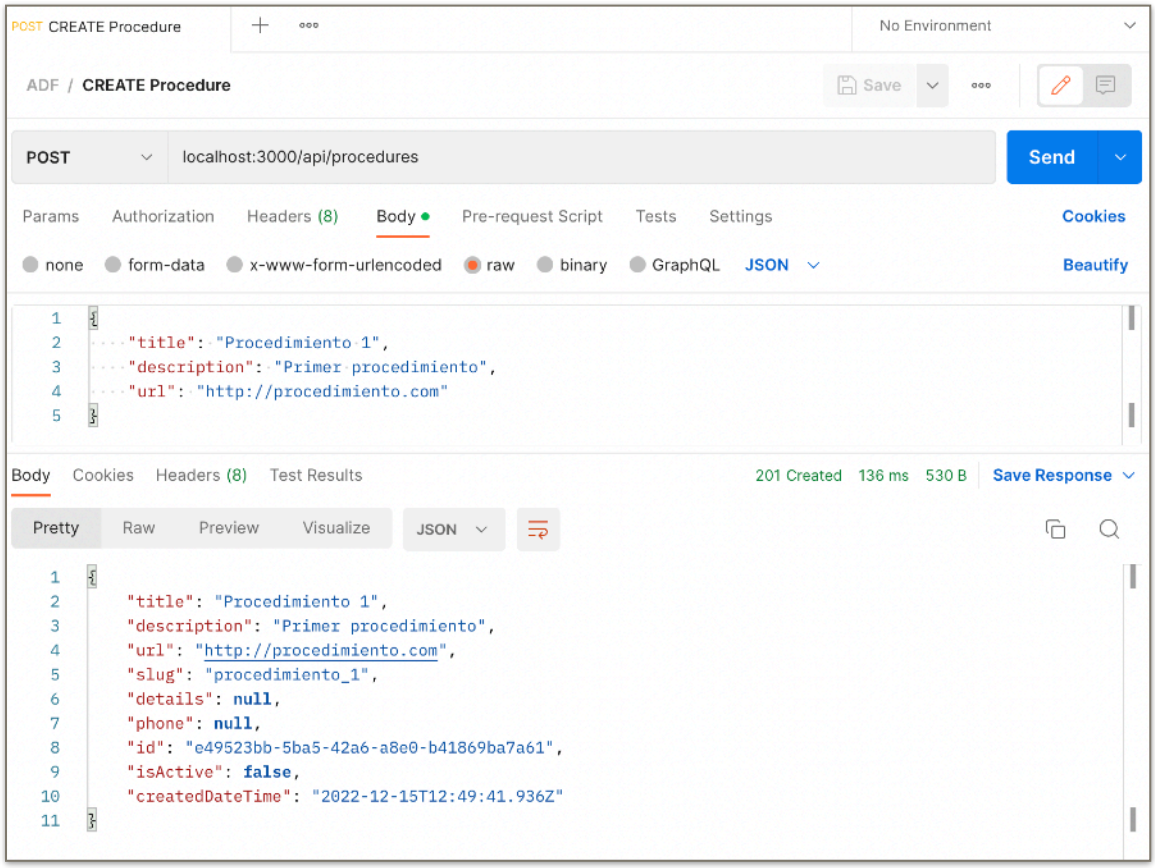
Ha llegado el momento de trabajar el CRUD de las tablas y generar los endpoints correspondientes. Con la configuración actual, si navegamos hasta localhost:3000/api/procedures recibiremos el mensaje configurado por defecto en el servicio, pasará lo mismo si empleamos los métodos GET y POST en Postman, y si añadimos un string en la dirección (localhost:3000/api/procedures/id) recibiremos también un mensaje con los métodos GET, PATCH y DELETE.



5.1 Respuesta en Postman de un método GET con la configuración por defecto de un CRUD

Empezamos el desarrollo con la inserción de datos, para lo cual desarrollaremos el método create del servicio correspondiente. Antes de llevarlo a cabo es importante implementar los DTO ya que nos servirán de referencia de cómo vamos a ver la inserción de los datos. El dto, las siglas en inglés de data transfer object, definirá cómo recibimos la petición del endpoint para pasársela al controlador. Indicaremos la información que esperamos, la opcional y la que se omitirá para ser generada automáticamente. Para ayudarnos en la tarea de validación utilizamos dos librerías externas: class-validator y class-transformer.

La creación se desarrolla entre el controlador y el servicio de cada recurso, para llevarlo a cabo empleamos un patrón repositorio que será el encargado de realizar las interacciones con la base de datos (procedureRepository). El método create será asíncrono, porque las relaciones con la base de datos son asíncronas, y dentro del mismo usaremos la estructura try-catch para poder manejar también los errores, que desarrollaremos mediante un logger en consola y un método privado en el servicio. Con todo ello podemos crear el endpoint de creación en Postman.



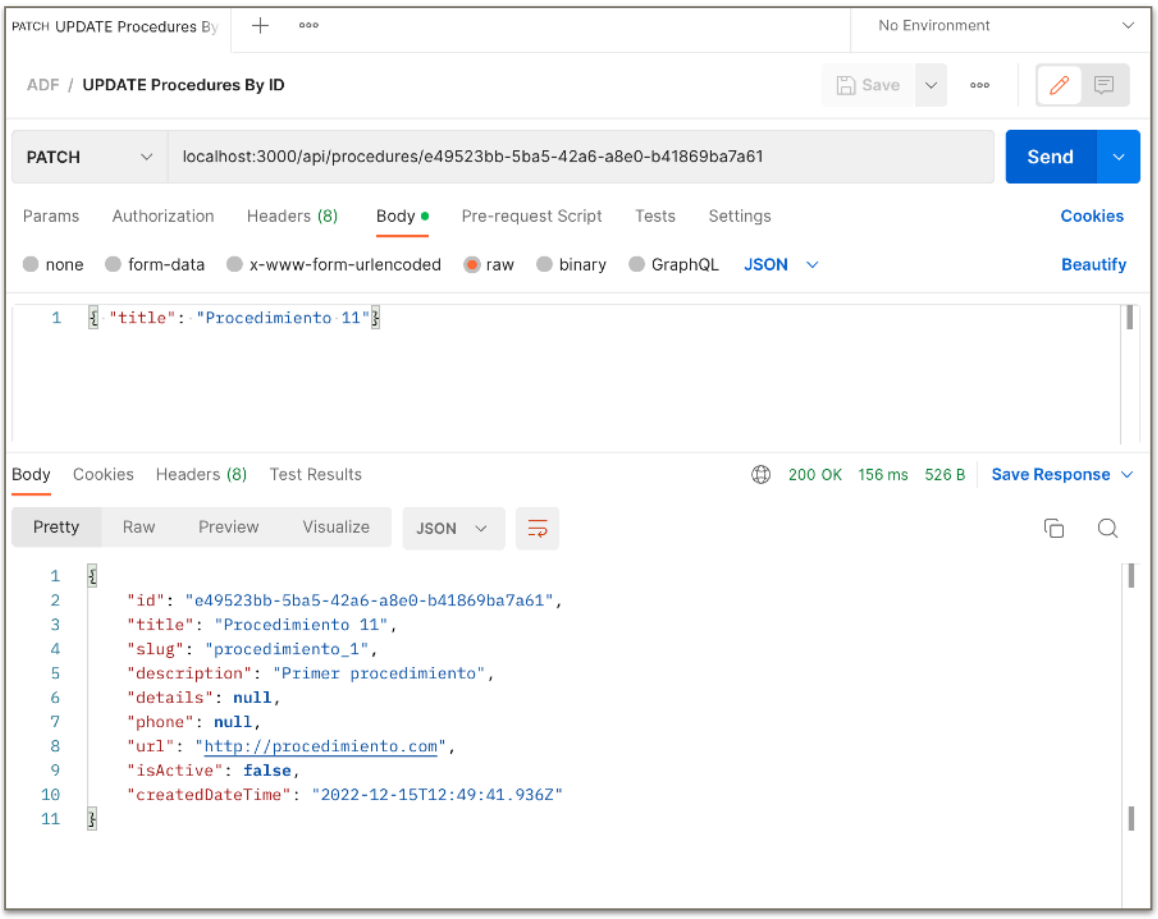
5.2 Respuesta en Postman del método Create en la entidad Procedure

A continuación se desarrolla, gracias al Repository, todos los métodos GET que tenga cada recurso, desde obtener todos, obtener todos los que conlleven alguna cualidad específica u obtener un producto por una característica concreta, que normalmente es el identificador único. En el último caso es importante recordar que la estructura por defecto

espera un identificador numérico, mientras que en nuestro caso hemos optado por identificadores de tipo UUID y necesitaremos adaptarlo al modo string.

El método de eliminación (remove) se crea relacionado con un método existente de búsqueda única para evitar duplicar la necesidad de comprobar que el objeto solicitado para la eliminación existe. Posteriormente, se elimina de la base de datos gracias al patrón repositorio.

A la hora de gestionar la actualización es importante tener presente que todos los campos pueden ser opcionales. Nest ya contempla esta situación, de hecho al crear el recurso no solo crea un DTO para la creación, también extiende otro para la actualización haciendo básicamente opcionales todas las opciones, de esta manera podemos pasar al método el id con los elementos modificados y no recibir el error que ocasionaría si fuese la creación.
El siguiente paso será la carga de archivos, en nuestro caso las imágenes de las instituciones (agencies). Para este fin utilizamos multer, que viene por defecto en Nest aunque requiere de la instalación de los tipos para el tipado estricto de TypeScript.



5.3 Respuesta en Postman del método Update de la entidad Procedure

Preparamos un nuevo recurso rest API para subir los archivos, aunque en este caso no necesitaremos implementar un CRUD completo ya que solo subiremos las imágenes y las buscaremos, y tampoco vamos a emplear entidades o DTOs.

Con ayuda de los decoradores de Nest podemos interceptar y cargar los archivos. Aunque para evitar que pueda cargarse cualquier tipo, necesitaremos validar que son exclusivamente los que necesitamos, es decir, imágenes. De esta manera el interceptar enviará la referencia a la validación y podrá dar el acceso o rechazar el archivo.

Por defecto los archivos cargados se guardan de manera temporal por lo que necesitamos buscar un espacio para almacenar las imágenes. En esta ocasión lo haremos en el propio filesystem del proyecto dentro de la carpeta static, y a su vez en otra llamada uploads. En un futuro, si la aplicación de desplegase para uso, sería interesante plantearse almacenar las imágenes en otro servidor ajeno a la API, como Cloudinary o AWS, y así evitar las vulnerabilidades de seguridad que pudiese tener al estar en el mismo árbol que la lógica del desarrollo. Precisamente por esta razón, y como pequeña medida de seguridad, no vamos a almacenar las imágenes en una carpeta pública, es preferible usar otro sistema, como la carpeta static, para tener mayor control y facilitar el archivo al cliente sin dar más acceso del necesario.

Por defecto el almacenamiento se produce con el mismo nombre que se recibe, de manera que si dos imágenes, aunque sean diferentes, se cargan en el sistema, la segunda sobreescribiría a la primera. Por esa razón se gestiona un helper que nos ayude a renombrarlas y aprovechamos para incorporar la extensión, que también se elimina por defecto. Para dar el nuevo nombre, y asegurarnos que será único se utiliza el sistema de identificación universal uuid, que también implementamos como forma de numerar los identificadores únicos de las tablas, aunque en este caso, para emplearlo fuera de ese contexto necesitamos añadir el paquete. Una vez instalado y configurado el helper mediante una callback, el sistema tendrá las herramientas necesarias para generar un nombre único a las imágenes y devolver la ruta en la que encontrarlas.

El siguiente paso a implementar es el módulo encargado de la gestión de usuarios y su autenticación. Este módulo cuestionó bastantes dudas en su implementación y se llegó a plantear la posibilidad de realizar dos recursos separados, uno para la autenticación y otro para la gestión de usuarios, finalmente se optó por uno solo ya que ambos se gestionan desde la misma entidad de usuario y las funciones de login y creación van estrechamente relacionadas entre sí.

En este caso, como trabajamos con contraseñas se implementa el paquete bcrypt para la encriptación de las mismas y evitar así que puedan verse comprometidas en algún momento. Incluso si accedemos directamente a los datos es imposible ver la contraseña, y este paquete nos facilita su manejo y cotejo.

Para llevar a cabo la autenticación, según las recomendaciones de la documentación oficial de Nest, empleamos la librería Passport y los paquetes necesarios para implementar la estrategia JWT.

Para configurar el jwt se crea con una interfaz y una estrategia de uso. Una vez implementados los métodos de creación y login con jwt creamos las rutas privadas. También implementamos un método para obtener el usuario desde la autenticación.

Generamos un guard y los decoradores personalizados necesarios para poder especificar qué usuarios tendrán acceso a determinados métodos, ya que decidimos en el inicio del proyecto que no todos los usuarios registrados contarían con los mismos privilegios.

Aprovechamos que contamos con los usuarios para ampliar las entidades de las instituciones y los procedimientos para poder indicar quién creo cada uno de los datos y quién es la última persona que los ha actualizado.

Uno de los problemas cuando se trabaja en desarrollo son los datos contenidos en la base de datos. La creación de muchos elementos con los que hacer pruebas puede ser muy tediosa, y en ocasiones puede que necesitemos eliminar algunos y perder la referencia relacional que guarden con otros elementos, algo que se contempla a la hora de eliminar algunos datos que guarden relación con otras tablas. En el caso de los procedimientos asociados a una institución, son eliminados si borramos esa institución, y en el caso de las referencias de autorías, se convierten en campos null.

Para facilitar una carga de datos completa de manera local, se implementa un nuevo módulo semilla, seed, que incorporará una serie de registros de forma automática a la base de datos con una simple llamada a su correspondiente endpoint.

Este método no cuenta con ningún tipo de requerimiento de autenticación ya que se trata, exclusivamente, de un método para el desarrollo de la aplicación porque cuenta con herramientas altamente destructivas de todos los datos de la base de datos. De hecho, si decidiese implementarse en un entorno en producción debería restringirse a muy pocos usuarios expertos que no pongan en riesgo el sistema.

El último paso implementado en el bakend ha consistido en la documentación de la API, realizada con ayuda del módulo Open API (Swagger) para Nest que se instala mediante la consola de comando y se configura según señala la documentación oficial.

Se puede acceder a ella en el url raíz del proyecto, mientras está en ejecución, que por defecto será localhost:3000/api.

Gracias a la documentación de Swagger se ha ampliado el modelo por defecto mediante decoradores como ApiTags, ApiProperty o ApiResponse.

5.2. Desarrollo del Frontend

a. Herramientas necesarias para el frontend durante la fase de desarrollo

Para llevar a cabo el frontend se ha empleado el framework de Angular, una herramienta utilizada en alguna de las asignaturas de este máster, al igual que el manejo del backend se ha utilizado el gestor de paquetes npm y en su mayor parte se ha utilizado la herramienta de Angular CLI para poder llevarse a cabo.

Si bien no es necesario para la maquetación de los templates, es necesario tener corriendo la aplicación del backend, o conectarse a ella en remoto, para poder llevar a cabo el proceso total de desarrollo de esta parte visual.

b. Cómo utilizar la aplicación en desarrollo

Son muy pocos los pasos necesarios para poder utilizar la fase en un proceso de desarrollo en local. En primer lugar tenemos que descomprimir el archivo del frontend en la carpeta que decidamos de nuestro equipo, o clonarlo si optamos por emplear la copia del repositorio en Github[9].

Mediante la consola utilizaremos la instrucción *npm install*, o la abreviada *npm i*, para hacer la instalación del proyecto y todas sus dependencias.

Si no recibimos ningún mensaje de error que requiera alguna actualización, podemos ejecutar el proyecto mediante la instrucción *ng serve*. Podemos usar el flag -o para que se abra automáticamente en el navegador o acceder a través de la barra de dirección al puerto 4200 por defecto, a menos que decidamos cambiarlo, y ya podemos hacer uso de la aplicación por completo.

c. Paquetes instalados

- Angular CLI: https://www.npmjs.com/package/@angular/cli
- Bootstrap widget: https://ng-bootstrap.github.io
- Font Awesome: https://www.npmjs.com/package/@fortawesome/angular-fontawesome
- Angular Toggle Button: https://www.npmjs.com/package/ng-toggle-button
- Angular File Uploader: https://www.npmjs.com/package/angular-file-uploader

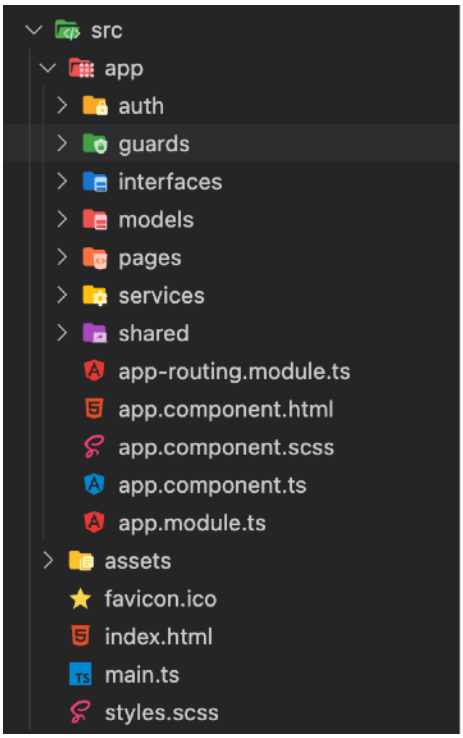
9 < https://github.com/saraemdev/adf.frontend >

d. Estructura de carpetas del proyecto

Existen múltiples opciones para poder ordenar un proyecto de Angular, depende mucho del tamaño del mismo pero lo normal es crear una estructura de carpetas más allá que generar todos los elementos en la carpeta inicial.

En esta ocasión las decisiones se han tomado en dos aspectos. La primera era intentar agrupar aquellos elementos que fuesen similares, por ejemplo todos los servicios juntos, todas las interfaces juntas… y en segundo término, como los componentes eran numerosos se buscó diferenciar en la estructura su utilidad, así encontramos agrupados los componentes que se comparten en otros como los que alojan la navegación o la página con el error no encontrado, por otro los que requieren condiciones de vulneración como el login, a medio camino entre la parte pública y la privada; y por último los elementos que componen la navegación, anidados entre el home y el dashboard:

```
> src
    > app
            > auth
                    > login
            > guards
            > interfaces
            > models
            > pages
                    > dashboard
                            > agencies
                                    > active-agency
                                    > agency-form
                                    > agency-list
                                    > my-agency
                            > procedures
                                    > active-procedure
                                    > my-procedure
                                    > procedure-form
                                    > procedure-list
                            > profile
                            > users
                                    > active-user
                                    > user-form
                                    > user-list
                    > home
                            > procedure
                            > procedures
            > services
            > shared
                    > footer
                    > header
                    > page-not-found
                    > sidebar
> assets
            > icons
            > img
```



5.4 Estructura de carpetas del proyecto de Angular

e. Componentes y servicios del proyecto

La función del componente es prácticamente la representación de una "página" o una parte de la representación en pantalla. Cuando creamos desde la consola un componente mediante la instrucción correspondiente *ng g c nombre_del_componente*, se generan por defecto cuatro archivos correspondientes al documento html, la hoja de estilos, el controlador y las pruebas unitarias.

Al igual que sucedía en el proyecto del backend se ha descartado la opción de las pruebas unitarias, y eliminado esos archivos al no ser necesarios. El proceso del framework Angular funciona de tal manera que el componente recibe, mediante el controlador, su inicio, este llama al modelo para recibir los datos que necesite, aunque no siempre se requieren más datos que el propio marcado, y devuelve esos datos a la vista para que se vea reflejado en la pantalla. Lo que es, como ya se explicó un perfecto modelo MVC.

Los componentes, a su vez pueden contener otros componentes, o mejor especificado, llamadas a otros componentes, bien desde sus controladores, bien desde etiquetas en las propias vistas.

La lógica del modelo se desarrolla en estos casos en los servicios, que son los encargados de comunicarse a su vez con la API y obtener datos de terceros.

En el desarrollo de esta aplicación se han generado varios componentes, detallados en la estructura de carpetas. Se ha intentado que sus nombres sean lo suficientemente descriptivos para no requerir de demasiada explicación, ni anotaciones a lo largo del código. Una técnica seguida también a lo largo del desarrollo de toda la parte lógica, nombrando funciones, variables… de la manera más descriptiva posible.

Merece mención aparte fijarse en alguno de los nuevo servicios que se han desarrollado en este proyecto. Además de crear un servicio por cada entidad de la base de datos para poder separar la lógica de cada elemento, se ha generado un servicio exclusivo para el login en la aplicación, garantizando mayor seguridad en el proceso, y otro para gestionar la carga de archivos, que en esta ocasión solo maneja las imágenes de las instituciones, y podríamos haberlo desarrollado en el servicio correspondiente, pero su lógica se separa por cuestiones de escalabilidad, ya que ese método podría servir para almacenar imágenes en otros servicios en el futuro.

Existen otros servicios más globales y necesarios para el correcto funcionamiento, en primer lugar se trabaja con un interceptor interceptor, que nos ayuda en la gestión de conocer si el usuario está registrado y añadir el token necesario en cada una de las peticiones al backend.

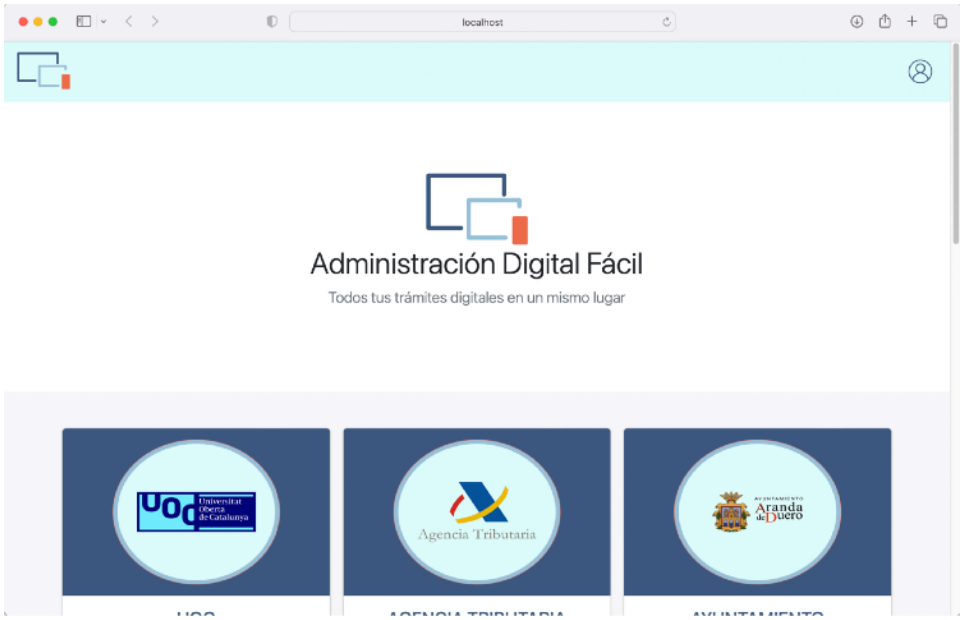
Por otro lado contamos con el servicio que gestiona el comportamiento de los menús en función de si el usuario está o no loqueado con éxito. Otro servicio comparte la gestión de

errores en las peticiones de los formularios de registro, mientras el otro gestiona la grabación, acceso y eliminación de los datos del login en el localstorage.

f. Breve descripción del desarrollo del backend

En primer lugar se configuró el proyecto con ayuda de Angular CLI y algunas de las dependencias externas que sabíamos inicialmente que necesitaríamos como Bootstrap, un framework CSS que nos ayuda notablemente en la realización de un marcado adaptable a cualquier tamaño de pantalla más fácilmente. Y el paquete de iconos Font Awesome.

En primer lugar se desarrollaron los componente home, footer y header junto al servicio agency para poder dar forma a la página de inicio y comprobar que tanto el framework como las dependencias y las llamadas al servidor se hacían de manera correcta.



5.5 Vista parcial de la página de inicio en modo local

El siguiente paso fue la creación del resto de componentes públicos que no requerían de autenticación y podían consumirse sin mayores problemas, para finalmente pasar a desarrollar los servicios necesarios para almacenar en localstorage, interceptar y la correspondiente guard para poder proteger las rutas que precisaban de autorización.

La parte final en desarrollar ha sido la parte del usuario registrado, poniendo especial empeño en mostrar los datos correctos en función del rol de usuario que los maneja, ya que si bien la aplicación de Nest no permitirá a un user borrar a un usuario, como desarrolladores en el frontend debemos evitar o minimizar al máximo que ese usuario pueda intentar acciones no permitidas o tener una mala experiencia de usuario, en especial porque muchos de esos errores no llegarán ni siquiera de manera visual si no los implementamos, y el usuario medio que utilizará la aplicación puede no haber abierto en su vida una consola para comprobar si hay algún error.

Una vez comprobados que todos los procesos funcionan, llega el momento de pasar a la parte de despliegue a producción con el fin último de alojar el proyecto en un servidor remoto para poder verse en la red.

Para finalizar los procesos en desarrollo, tanto en Nest como en Angular será suficiente con pulsar control+C en el teclado.

5.3. Proyecto en la red

a. Despliegue a producción

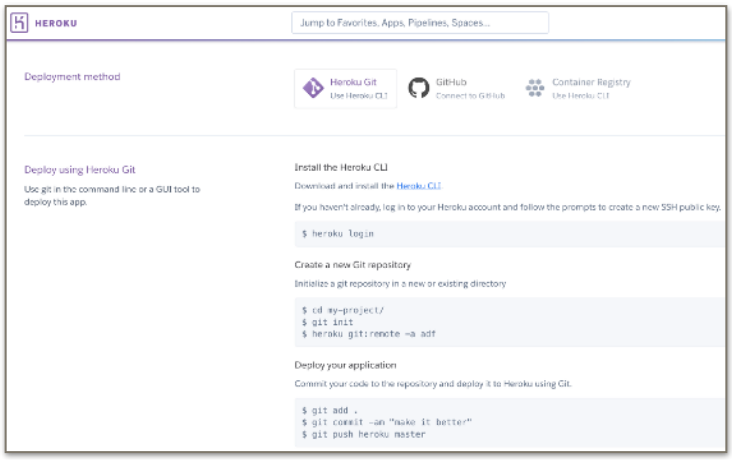
Una vez finalizado el proceso de desarrollo necesitamos hacer algunos cambios en algunas variables para poderlo desplegar a producción y que nos generen los proyectos sendas carpetas dist que podríamos alojar en un servidor remoto para su uso.

Para llevar a cabo el proceso de producción es necesario ejecutar el comando npm run build y ng build en los correspondientes proyectos de backend y frontend. Aunque en este caso primero se realizó el proceso con el backend para configurar los datos del nuevo servidor en las peticiones de Angular.

b. Alojando en Heroku y Netlify

En un proyecto profesional rara vez se suelen alojar conjuntamente las dos partes de un proceso, de hecho incluso el servicio de las imágenes y archivos se puede realizar en un tercer alojamiento. Por esta razón se ha decidido hacer la prueba de alojar en distintos servidores comerciales en lugar de utilizar el proporcionado por la UOC, de manera que se han puesto en práctica herramientas de terceros habituales en entornos profesionales actuales.

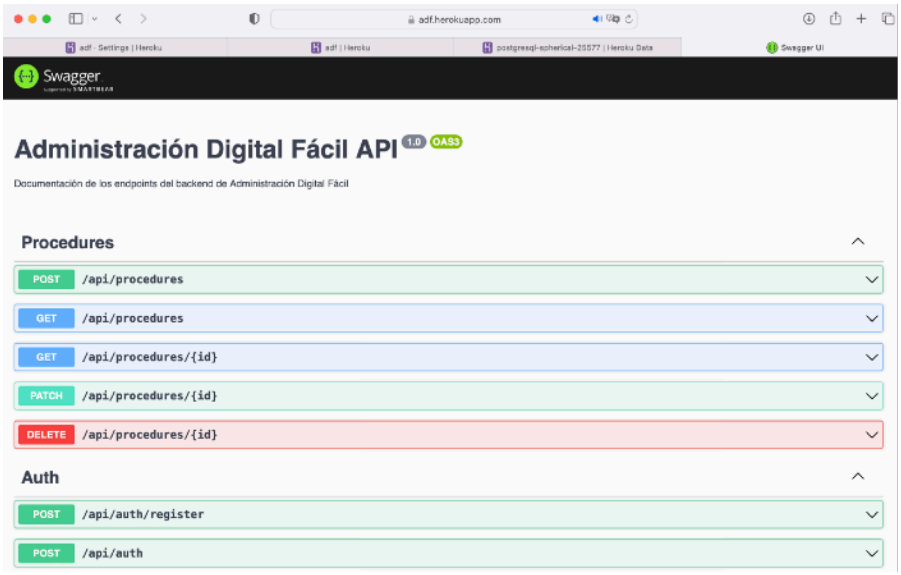
En primer término se instaló la CLI de Heroku para subir todo el proyecto del backend y la instalación mediante el mismo fue tan sencilla como seguir las indicaciones de su propia página web, mediante sencillas instrucciones muy similares a las empleadas para hacer copias de seguridad en Git.



5.6 Instrucciones para el despliegue mediante la CLI de Heroku

El siguiente paso fue establecer las variables necesarias para la conexión a la base de datos de Heroku Postgres, que son las mismas que se definieron en las variables locales (archivo .env) pero con los valores que nos facilita el servidor.

Tras este proceso el resultado de ejecutar la aplicación nos devolvió el error 404 que recibíamos al abrir la ruta raíz en local, ya que por defecto nuestra aplicación cuenta con un prefijo api. Y tras completar la dirección a < https://adf.herokuapp.com/api > podíamos acceder a la documentación del proyecto en línea sin más problema.



5.7 Vista de la documentación de la API de la aplicación en la red

A continuación se cambiaron las rutas de las peticiones en el frontend y se comprobó que podíamos navegar sin mayor dificultad a través del alojamiento en la web, con un único inconveniente en la ruta de las imágenes que generaba el seed por defecto. Para poder arreglarlo, y tras modificar su acceso a través de un usuario registrado, se hizo una segunda carga para mostrar correctamente los documentos y para evitar que una persona externa pudiese ejecutar el seed.

Llegados al momento en el que se siguió el mismo recorrido para alojar la parte pública de Administración Digital Fácil a través del CLI de Netlify. En este caso, no era necesaria la configuración de variables, simplemente indicar a Netlify la carpeta en la que se alojaba nuestro proyecto: dist/frontend.

c. Paquetes instalados

- Heroku CLI: https://www.npmjs.com/package/heroku
- Netlify CLI: https://www.npmjs.com/package/netlify-cli

d. Alojamiento de los repositorios en GitHub

Uno de los resultados que se especificó al inicio del proyecto fue el de alojar los repositorios correspondientes en GitHub de forma que puedan consultarse públicamente por parte del tribunal de este proceso o de las personas interesadas en conocer el proyecto que tengan acceso a esta memoria.

El proceso de creación de los repositorios de GitHub funciona mediante una serie de instrucciones git que el propio repositorio nos facilita cada vez que creamos uno nuevo, pero antes de llevarlo a cabo se modificaron algunos detalles tales como las contraseñas y nombres del archivo seed para evitar vulnerabilidades por despiste, así como los enlaces de Angular a las rutas de Heroku, volviendo a utilizar el puerto 3000 de localhost.

- Repositorio del backend: https://github.com/saraemdev/adf.backend
- Repositorio del frontend: https://github.com/saraemdev/adf.frontend

6. Resultados

6.1. Conclusiones y trabajos futuros

Una vez obtenidos los resultados, ¿qué conclusiones se extraen?

La primera conclusión alcanzada es la de un cierto grado de satisfacción al conseguir cumplir muchos de los objetivos propuestos e incluso alguno más de los imaginados a pesar de los numerosos problemas que han ido surgiendo tanto técnicos como, mayoritariamente, ajenos al proceso de desarrollo y que han mermado notablemente tiempo de dedicación al mismo.

El aprendizaje de Nest ha sido realmente enriquecedor y relativamente sencillo de aprender con poca más ayuda que la de la documentación oficial y algún tutorial. Es cierto que se basa en Angular, que era un framework utilizado a lo largo de dos asignaturas del máster, pero incluso me atrevo a afirmar que aprender Nest me ha hecho comprender mejor Angular.

Estos resultados, ¿son los esperados? ¿Han sido sorprendentes? ¿Por qué?

A nivel de productos de desarrollar y entregar se han alcanzado los mínimos deseables, si bien es cierto que la mayoría tienen un amplio rango de mejora no por ello carecen de valor.

En el nivel de curiosidad y aprendizaje debo reconocer que me ha sorprendido gratamente el uso, no solo de Nest y su fácil conexión con Angular para poder realizar aplicaciones interesantes, también estoy deseando poder profundizar en otras herramientas como Docker en el futuro.

¿Hemos alcanzado todos los objetivos? Si la respuesta es negativa, ¿por qué?

En el sentido más estricto podríamos decir que sí, hemos alcanzado todos los objetivos en cuanto a que hemos conseguido desarrollar una aplicación funcional completa que aúna los requisitos fijados en el inicio.

Desde un punto de vista más perfeccionista, se podría haber documentado la aplicación mejor, ya que Swagger ofrece más posibilidades de las empleadas y se podría haber refactorizado más el código en Angular, que apenas llega a estar completo en todas sus funciones.

¿Se ha seguido la planificación?

Se ha intentado ajustar al máximo a la planificación, no obstante no se ha podido seguir de manera estricta y algunos hitos han tenido que ser retrasados o directamente no realizados a causa de imprevistos no contemplados suficientemente.

¿La metodología prevista ha sido suficientemente adecuada?

La metodología creo que ha sido la más apropiada dado que hablamos de un equipo unipersonal y muchas veces no se podía continuar a otra tarea sin resolver una anterior.

¿Ha sido necesario introducir cambios para garantizar el éxito del trabajo? ¿Por qué?

La primera opción que surge es el exceso de optimismo. Afrontar un proyecto tan desconocido para mí hizo plantear quizá más de lo que el tiempo podía ofrecer, especialmente teniendo en cuenta factores externos de índole personal que han impactado en el resultado de falta de tiempo.

Los cambios estructurales básicos han sido la reducción al máximo de la fase de pruebas, detección de bugs y las pruebas con usuarios reales. Además algunos factores como la documentación podría haber sido más extensa y el manejo de errores en la parte del cliente más elaborado.

De los impactos previstos en 1.3, evaluar si se han mitigado o logrado

Dimensión de sostenibilidad.

Es muy difícil poder evaluar este apartado ya que el impacto en sostenibilidad mediante la modernización de tecnologías es una carrera a largo plazo en la que una aplicación como la desarrollada en este trabajo final tendría un mínimo impacto positivo en el conjunto global y sería muy precipitado afirmar rotundamente que dicho impacto se ha logrado.

Dimensión de comportamiento ético y de responsabilidad social.

En este caso el impacto positivo se puede evaluar con mayor claridad ya que las personas encuestadas dejaban clara la necesidad de promover una inclusividad como la que plantea este proyecto.

Dimensión de diversidad, género y derechos humanos.

La sencillez de uso de la aplicación en su uso general hace que podamos seguir pensando en un impacto positivo a la hora de reducir la desigualdad por cuestiones especialmente económicas y formativas.

Si han aparecido impactos no previstos, evaluar cómo se han mitigado

A la hora de hacer un saludo generalista en el acceso al parte registrada se ha optado por un Bienvenid@, algo que a simple vista podría ser una mitigación en el impacto negativo que tendría en una perspectiva de género. Sin embargo, el uso de la arroba como medida para decir Bienvenido y Bienvenida no solo excluye al resto de géneros si no que dificulta el acceso a personas con capacidades diferentes como aquellos que emplean lectores de pantalla y que no podrían resolver cómo denominar la palabra.

Las líneas de trabajo futuro que no han podido explorarse en este trabajo y han quedado pendientes

En primer lugar se debería implementar un buscador de procedimientos, además podemos utilizar perfectamente el endpoint que nos devuelve todos ellos con lo que solo sería necesario desarrollar la lógica en la parte de cliente.

En esa misma línea se podría implementar la opción de ordenación de las tablas, así como su filtrado. Ampliar entidades para dotar de imágenes a los procedimientos o detalles de contacto a las instituciones podría ser otra opción.

En la parte de la API se podría mejorar la documentación, y en ambos lugares la refactorización, especialmente en Angular donde hay bastante código repetido en varios componentes.

Por último las líneas que quedan pendientes son la profundización en Docker, Postgres y otras tecnologías como React o Vue con las que intentar replicar el ejemplo.

7. Glosario

Analfabetismo digital:
Se conoce así al nivel de desconocimiento en tecnologías de la información que impide a la persona que lo sufre el acceso e interacción con estas herramientas.

API REST:
También conocida como REST API o API. Se trata de un conjunto de definiciones y protocolos que diseñan el software de las aplicaciones. Hace funciones intermedias entre los datos y el cliente que lo solicita.

Asíncrono:
Un tipo de comunicación por el que el intercambio de información no se realiza inmediatamente, sino cuando se consigue procesar. Los intercambios de información que impactan en bases de datos son siempre asíncronos.

Autenticación (login):
Acreditación, por medios electrónicos de la identidad y de la autoría de sus actos en un trámite electrónico.

Backoffice:
Conjunto de herramientas que facilitan la gestión interna de una empresa, en términos informáticos se emplea como sinónimo de Sistema de gestión de contenidos. Ver CMS.

BD:
Siglas de Base de Datos. Programa informático capaz de almacenar datos para futuras consultas y modificaciones de los mismos.

Brecha digital:
Desigualdad de acceso y uso de las tecnologías de la información entre grupos sociales, normalmente pronunciada en base a criterios económicos, geográficos, género y edad.

Certificado electrónico:
Documento firmado electrónicamente por un prestador de servicios de certificación que vincula unos datos de verificación de firma a un firmante y confirma su identidad.

Callback:
Son funciones que se pasan a otras como argumentos para realizar alguna rutina dentro de la primera función.

CMS:
Siglas de Content Management System (Sistema de gestión de contenidos). Consiste en el sistema que permite administrar dinámicamente contenidos digitales, normalmente webs, blogs, apps… de una forma fácil para el usuario que aporta la información

CRUD:
Son las siglas de Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar). Es la agrupación de las cuatro acciones básicas que sirven para gestionar la información que se almacena en desarrollo.

Dependencias:
Se trata de una aplicación o biblioteca necesaria para que un programa o una función pueda funcionar correctamente. En los frameworks empleados en esta práctica es muy habitual encontrar inyecciones de dependencias en muchos constructores.

DTO:
Son las siglas de Data Transfer Object (Objeto de transferencia de datos). Consiste en un patrón que tiene como finalidad la creación de un objeto con los atributos que pueden ser enviados o recuperados en una única llamada. En el caso de Nest y Angular definen cómo se reciben los datos de un endpoint para pasarlas al controlador.

Endpoint:
Las URL de una API que responden a una definición de petición de la misma. Las más habituales son de tipo GET, POST, PUT, PATCH y DELETE.

Flag:
Conocido en español también como bandera, es una variable lógica que nos permite indicar cambios. A la hora de crear contenido mediante la consola es muy habitual su uso para hacer alguna referencia rápida como que no instale alguna cosa por defecto o que abra la pantalla del navegador al terminar la acción.

Framework:
Conjunto de herramientas que permite agilizar los procesos de desarrollo al evitar escribir código de manera repetitiva y que facilita, en gran medida las buenas prácticas, consistencia y mantenimiento del código. Los más habituales son los que usan el paradigma MVC, pero también podemos encontrar frameworks de estilo como Bootstrap o Tailwind

Gestor de paquetes:
Colección de herramientas que ayudan a automatizar procesos de instalación, actualización, configuración y eliminación, en la creación de un producto digital.

Guard:
En Angular, son interfaces que permiten determinar si una ruta puede o no cargarse en función de parámetros necesarios como por ejemplo un usuario registrado.

Identificador UUID:
Son las siglas de Universally Unique Identifier (Identificador Único Universal). Son identificadores complejos creados por una máquina en función de numerosos parámetros para evitar duplicidades.

Interceptor:
En Angular, son un tipo de servicio especial que aportan un mecanismo para gestionar las solicitudes y respuestas http.

JWT:
Son las siglas de JSON Web Token. Un estándar que establece transmisión de información entre uno o más campos de forma segura y verificada, ya que cuentan con firma virtual.

Librería:
Conjunto de archivos de código que se utilizan durante la fase de desarrollo. El objetivo es proporcionar hechas las funcionalidades más comunes sin necesitar de repetirlas una y otra vez.

Localstorage:
Es una propiedad que puede gestionar información dentro del navegador. Puede almacenar, recuperar y eliminar la información incluso entre pestañas del navegador y guardar más contenido que las antiguas cookies.

Patrón repositorio:
Consiste en un patrón de diseño que aísla la capa de datos del resto de la aplicación. En el caso de Nest, es el encargado de realizar las interacciones con la base de datos.

Postgres:
Forma habitual de referirse a PostgreSQL, un sistema de gestión de bases de datos relacional orientado a objetos de código abierto.

Pruebas unitarias:
Son test que ponen a prueba el correcto funcionamiento de unidades pequeñas del código.

SPA:
Siglas de Single Page Application (Aplicación de página única). Consiste en una aplicación web donde todas las pantallas se muestran en la misma página sin necesidad de recargar el navegador en cada solicitud.

Sede electrónica:
Sitio web que está a disposición de la ciudadanía para acceder a los servicios y trámites electrónicos de la administración pública encargada de su gestión y administración.

TypeORM:
Librería empleada para la creación y manipulación de datos sin necesidad de emplear lenguaje SQL.

URL:
Son las siglas de Uniform Resource Locator (Localizador Uniforme de Recursos). Consiste en la dirección única y específica que cada recurso tiene en la red.

Usabilidad:
La cualidad de una web para poder ser utilizada de manera satisfactoria y correcta por el tipo de usuarios que forman su público objetivo.

Ventanilla única:
Punto de contacto en el que las Administraciones Públicas garantizan la información relativa al ejercicio de los servicios y donde pueden llevar a cabo todos los trámites para el acceso y ejercicio de sus actividades de servicios.

Wireframe:
En diseño web, consiste en un boceto de la representación visual de la página empleado en las primeras etapas del proceso de creación. Estos diseños pueden ir desde meros trazados a prototipos de alta fidelidad con un gran lujo de detalle.

8. Bibliografía

8.1. Libros

Albuera Reverté, F. (2021) *Desarrollo front-end avanzado*. Apuntes. Barcelona: UOC

Bampakos, A. (2021) *Angular projects*. Segunda edición. Birmingham: Packt Publishing. ISBN 9781800205260

Correa, D. y Lin, G. (2022) *Practical Nest.js. Develop clean MVC web Applications*. Primera edición. Autopublicación en formato digital por Daniel Correa

Varios autores. (2018) *Nest.js: A Progressive Node.js framework*. California: Bleeding Edge Press. ISBN 9781939902627

8.2. Artículos

European Parliament. *Consejo Europeo de Santa Maria da Feira. Conclusiones de la Presidencia*, Junio 2000. Edición digital [última consulta: noviembre 2022]
< https://www.europarl.europa.eu/summits/fei2_es.htm >

Varela Ferrío, José. *La brecha digital en España. Estudio sobre la desigualdad postergada*, 2015. Edición digital en PDF [última consulta: noviembre 2022] *UGT*
< http://portal.ugt.org/Brecha_Digital/BRECHADIGITAL_WEB.pdf >

8.3. Webgrafía

Angular. *Documentación* [última consulta: enero 2023]
< https://angular.io/docs >

Beatriz Allas Miguelsanz. (2017). *Los 10 principios de usabilidad de Jakob Nielsen: be user friendly* - Profile [última consulta: noviembre 2022]
< https://profile.es/blog/los-10-principios-de-usabilidad-web-de-jakob-nielsen/ >

Bootstrap. *Documentación* [última consulta: enero 2023]
< https://getbootstrap.com/docs/5.2/getting-started/introduction/ >

Castris Technology. (2020). *TablePlus, un antes y un después, para trabajar con bases de datos remotas* [última consulta: diciembre 2022]
< https://castris.com/tableplus-un-antes-y-un-despues-para-trabajar-con-bases-de-datos-remotas/ >

Christian Janker. (2020). *RxJS heads up: toPromise is being deprecated*
[última consulta: enero 2023]
< https://indepth.dev/posts/1287/rxjs-heads-up-topromise-is-being-deprecated >

Data Pilot. (2019). *PostgreSQL timestamp vs timestamptz* - Object Rocket [última consulta: noviembre 2022]
< https://kb.objectrocket.com/postgresql/postgresql-timestamp-vs-timestamptz-616 >

Desarrollo Web. (2020). *Qué es MVC* [última consulta: noviembre 2022]
< https://desarrolloweb.com/articulos/que-es-mvc.html >

Docker. *Documentación* [última consulta: diciembre 2022]
< https://docs.docker.com/ >

Dominicode. (2020). *Cómo publicar tu aplicación de Angular en Netlify* [última consulta: enero 2023]
< https://youtu.be/3VVPk7YYpho >

Eduardo Zepeda. (2020). *¿Qué es Docker y para qué sirve? Explicación* [última consulta: diciembre 2022] < https://dev.to/prox_sea/que-es-docker-y-para-que-sirve-explicacion-5h2n >

Ese Sunday. (2022). *Using TypeORM's QueryBuilder in NestJS* [última consulta: diciembre 2022] < https://blog.logrocket.com/using-typeorms-querybuilder-nestjs/ >

Font Awesome. *Documentación* [última consulta: diciembre 2022]
< https://fontawesome.com/docs/web/use-with/angular >

Gitbook. *Documentación de TypeORM* [última consulta: diciembre 2022]
< https://orkhan.gitbook.io/typeorm/docs >

GitHub Docs. *Creación de un token de acceso personal* [última consulta: diciembre 2022]
<https://docs.github.com/es/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

Gokul Budha. (2022). *Interceptors in Angular* [última consulta: enero 2023]
< https://medium.com/@gokulbudha/interceptors-in-angular-aee9b247d4c5 >

Heroku Del Center. *Deploying Node.js Apps on Heroku* [última consulta: enero 2023]
< https://devcenter.heroku.com/articles/deploying-nodejs >

Instituto Nacional de Estadística. (2020). *Encuesta sobre equipamiento y uso de tecnologías y comunicación en los hogares 2020* [última consulta: noviembre 2022]
< https://www.ine.es/dynt3/inebase/es/index.htm?padre=6899&capsel=6912 >

JWT. *Debugger* [última consulta: diciembre 2022]
< https://jwt.io/ >

Klerith. (2022). *Password-property-dto.ts* [última consulta: enero 2023]
< https://gist.github.com/Klerith/c77edd9d86174aa68efbfddd21f053a2 >

Krishna Damaraju. (2022). *NestJS and 'class validator' cheat sheet* - Dev
[última consulta: noviembre 2022]
< https://dev.to/sarathsantoshdamaraju/nestjs-and-class-validator-cheat-sheet-13ao >

Luis Aguilar. (2018). *Angular: Autenticación usando interceptors*
[última consulta: noviembre 2022]
< https://medium.com/@insomniocode/angular-autenticación-usando-interceptors-a26c167270f4 >

Manisha Jena. (2022). *PostgreSQL Varchar, Text and Character Data Types Made Easy | A 101 Guide* - Hevo [última consulta: noviembre 2022]
< https://hevodata.com/learn/postgresql-varchar/ >

Manuel Ortega Carcamo. (2021). *Iniciando con typeOrm para Nodejs* [última consulta: diciembre 2022] < https://dev.to/mortegac/iniciando-con-typeorm-para-nodejs-1dco >

MDN Web Docs. *La desestructuración* [última consulta: noviembre 2022]
< https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment >

Mi blog técnico. (2012). *Desarrollo de aplicaciones con el framework MVC de .NET* [última consulta: noviembre 2022]
< https://miblogtecnico.wordpress.com/2012/07/16/desarrollo-de-aplicaciones-con-el-framework-mvc-de-net-i/ >

Michael Karén. (2019). *Esperando lo inesperado - Buenas prácticas para el manejo de errores de Angular* [última consulta: enero 2023]
< https://michael-karen.medium.com/esperando-lo-inesperado-buenas-prácticas-para-el-manejo-de-errores-en-angular-dc578da68ef9 >

Miguel Ángel Álvarez. (2021). *Introducción a NestJS* [última consulta: diciembre 2022]
< https://desarrolloweb.com/articulos/introduccion-nestjs >

Miguel Ángel Álvarez. (2020). *Relación de uno a muchos con TypeORM y NestJS* [última consulta: diciembre 2022]
< https://desarrolloweb.com/articulos/relacion-uno-a-muchos-typeorm-nest >

Miguel Ángel Álvarez. (2020). *Servicios en Angular* [última consulta: diciembre 2022]
< https://desarrolloweb.com/articulos/servicios-angular.html >

Miguel Ángel Álvarez. (2022). *TypeORM en Nest Framework* [última consulta: diciembre 2022] < https://desarrolloweb.com/articulos/typeorm-nest-framework >

Naciones Unidas. *Objetivos de Desarrollo Sostenible* [última consulta: noviembre 2022]
< https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ >

Nest. *Documentación* [última consulta: diciembre 2022]
< https://docs.nestjs.com/ >

Nestjs-query. *Types* [última consulta: diciembre 2022]
< https://doug-martin.github.io/nestjs-query/docs/graphql/types/ >

Netlify Docs. *Gea started with Netlify CLI* [última consulta: enero 2023]
< https://docs.netlify.com/cli/get-started/ >

NgBootstrap. *Documentación* [última consulta: enero 2023]
< https://ng-bootstrap.github.io/ >

PostgreSQL. *Documentación* [última consulta: diciembre 2022]
< https://www.postgresql.org/docs/14/index.html >

Programación en español. (2022). *Git desde cero* [última consulta: diciembre 2022]
< https://youtu.be/B5BALozzzL4 >

Redacción KeepCoding. (2022). *¿Que es Postman?* [última consulta: diciembre 2022]
< https://keepcoding.io/blog/que-es-postman/ >

Rubén Aguilera Díaz-Heredero. (2020). *Creación de un CRUD con API REST y PostgreSQL en Nest* [última consulta: diciembre 2022]
< https://dev.to/raguilera82/creacion-de-un-crud-con-api-rest-en-nestjs-4ao3 >

RxJS. *Documentación* [última consulta: enero 2023]
< https://rxjs.dev/deprecations/to-promise >

Saurabh Dashora. (2021). *Simple Guide to NestJS Swagger Operations* [última consulta: diciembre 2022] < https://progressivecoder.com/a-simple-guide-to-nestjs-swagger-operations/ >

Simon Gausmann. (2019). *NestJS, TypeORM and PostgreSQL - full example development and project setup working with database migrations* [última consulta: diciembre 2022]
< https://medium.com/@gausmann.simon/nestjs-typeorm-and-postgresql-full-example-development-and-project-setup-working-with-database-c1a2b1b11b8f >

Stackoverflow. (2018). *Angular routerLinkActive for multiple paths* [última consulta: enero 2023] < https://stackoverflow.com/questions/49114399/angular-routerlinkactive-for-multiple-paths >

Suffle for Bootstrap. *All Bootstrap CSS classes* [última consulta: diciembre 2022]
< https://bootstrapshuffle.com/es/classes >

Swagger. *Documentación* [última consulta: diciembre 2022]
< https://swagger.io/docs/ >

TechOverflow. *How to fin Angular HttpClient to Promise() deprecated (rxjs)*
[última consulta: enero 2023]
< https://techoverflow.net/2022/01/17/how-to-fix-angular-httpclient-topromise-deprecated-rxjs/ >

TypeORM. *Documentación* [última consulta: diciembre 2022]
< https://orkhan.gitbook.io/typeorm/ >

TypeORM. *Documentación* [última consulta: diciembre 2022]
< https://typeorm.io/ >

TypeScript. *Documentación* [última consulta: diciembre 2022]
< https://www.typescriptlang.org/docs/ >

VManuelPM. (2020). *¿Cómo implementar un interceptor en Angular? Envío de token en la cabecera HTTP* [última consulta: enero 2023]
< https://amoelcodigo.com/interceptor-angular/ >

Víctor Cuervo. (2019). *¿Qué es Postman?* - AiT [última consulta: noviembre 2022]
< https://www.arquitectoit.com/postman/que-es-postman/ >

Web.dev. *User interface patterns* [última consulta: noviembre 2022]
< https://web.dev/learn/design/ui-patterns/ >

8.4. Material gráfico de terceros

Andrew Neel. *Stone Mountain State Park* [última consulta: enero 2023]
< https://unsplash.com/es/fotos/aebPbwAWjDs >

Colton Duke. *Brooklyn Bridge* [última consulta: enero 2023]
< https://unsplash.com/es/fotos/UExx0KnnkjY >

Dirección General de Tráfico. *Logotipo de la DGT* [última consulta: diciembre 2022]
< https://es.wikipedia.org/wiki/Archivo:DGT_logo.svg >

Instituto Nacional de la Seguridad Social. *Logotipo del INSS* [última consulta: diciembre 2022] < https://es.m.wikipedia.org/wiki/Archivo:Logo_INSS.svg >

Junta de Castilla y León. *Logotipo de la Junta de Castilla y León* [última consulta: diciembre 2022] < https://es.wikipedia.org/wiki/Archivo:Logotipo_de_la_Junta_de_Castilla_y_León.svg >

Junta de Castilla y León. *Logotipo del ECYL* [última consulta: diciembre 2022]
< https://es.wikipedia.org/wiki/Archivo:Logo_ECYL.png >

Junta de Castilla y León. *Manual de identidad corporativa* [última consulta: diciembre 2022] < https://www.jcyl.es/web/jcyl/Gobierno/es/Plantilla100DetalleFeed/1246464876027/Publicacion/1164017345952/Redaccion >

Ministerio de Hacienda y Función Pública. *Logotipo de la Agencia Tributaria* [última consulta: diciembre 2022] < https://commons.wikimedia.org/wiki/File:Agencia_Tributaria.svg >

Ministerio del Interior. *Logotipo de la Policía Nacional* [última consulta: diciembre 2022]
< https://es.wikipedia.org/wiki/Archivo:Logotipo_de_la_Policía_Nacional_de_España_(Tipográfico).svg >

Ministerio de Trabajo y Economía Social. *Logotipo del SEPE* [última consulta: diciembre 2022] < https://es.m.wikipedia.org/wiki/Archivo:Logo_SEPE.svg >

SACYL. *Logotipo del SACYL* [última consulta: diciembre 2022]
< https://es.wikipedia.org/wiki/Archivo:Sacyl.svg >

SACYL. *Manual de Identidad Corporativa de la Gerencia Regional de Salud* [última consulta: diciembre 2022] < https://www.saludcastillayleon.es/institucion/es/identidad-corporativa/manual-identidad-corporativa-gerencia-regional-salud >

UOC. *Libro de estilo de la UOC* [última consulta: diciembre 2022]
< https://www.uoc.edu/portal/es/llibre-estil/descarregues/logotips/index.html >

9. Anexos

9.1. Investigación sobre el panorama actual

A lo largo de dos meses, desde el 16 de octubre hasta el 16 de diciembre, se planteó la posibilidad de llevar a cabo una encuesta pública en internet a través de la cual tomar conciencia del conocimiento y la relación digital que los usuarios tienen con la Administración Pública.

A continuación se pueden consultar las preguntas y respuestas obtenidas. Todas las preguntas, con excepción de la opinión final, son obligatorias.

a. Cuestionario realizado

El enlace para acceder a la encuesta es: < https://forms.gle/geAkm7Lmr4dnsH4M8 >

Sección 1: Introducción explicativa sobre el estudio

Sección 2: Perfil de usuario

¿Cuántos años tienes?
- Menos de 20 años
- Entre 20 y 29
- Entre 30 y 44
- Entre 45 y 59
- Entre 60 y 76
- Más de 76

¿Con qué género te identificas?
- Femenino
- Masculino
- No binario
- Prefiero no decirlo
- Otro

¿Desde qué Comunidad Autónoma realizas esta encuesta?
- Andalucía
- Aragón
- Asturias
- Baleares
- Canarias
- Cantabria
- Castilla-La Mancha
- Castilla y León
- Cataluña
- Ceuta
- Comunidad Valenciana
- Extremadura
- Galicia
- La Rioja

- Madrid
- Melilla
- Murcia
- Navarra
- País Vasco

¿Cómo es la población en la que resides actualmente?
- Menos de 10.000 habitantes
- Entre 10.000 y 25.000
- Entre 25.000 y 50.000
- Entre 50.000 y 100.000
- Entre 100.000 y 250.000
- Más de 250.000 habitantes

¿Cuál es tu máximo nivel de estudios alcanzado?
- Sabe leer y escribir
- Estudios primarios incompletos
- Estudios primarios (Graduado escolar)
- Secundarios incompletos
- Bachillerato o Ciclo Formativo
- Grado universitario
- Máster universitario y/o Doctorado

¿Tienes acceso a internet en casa?
- Sí
- No

¿A cuál de los siguientes dispositivos electrónicos tienes acceso?
- Teléfono fijo
- Teléfono móvil sin conexión a internet
- Smartphone (móvil con conexión a internet con datos o Wi-Fi)
- Tablet
- Ordenador portátil
- Ordenador de sobremesa
- Smart TV (Televisión con conexión a internet)
- Consola de videojuegos con acceso a internet
- Ninguna de las anteriores

¿Utilizas alguna de las siguientes redes sociales?
- WhatsApp
- Instagram
- Facebook
- Tik tok
- Twitter
- YouTube
- Telegram
- Discord

Sección 3: Administración electrónica

¿Tienes certificado electrónico o acreditación equivalente digital?
- Sí
- No

¿Sabes encontrar el CIP o CIPA de tu tarjeta sanitaria para comprobar tus datos médicos digitalmente?

- Sí, siempre realizo las cuestiones médicas telemáticamente
- Sí, aunque suelo usar la web o la app solo algunas veces
- No. Siempre prefiero acudir en persona o llamar por teléfono

En los últimos 12 meses, ¿has solicitado cita para renovar tu DNI o Pasaporte digitalmente?

- Sí. Aboné las tasas telemáticamente
- Sí. Aboné las tasas en el momento de realizarlo
- No. Necesité ayuda para realizarlo por otro medio
- No. No he tenido necesidad pero sabría hacerlo
- No conocía esa posibilidad

En los últimos 12 meses, ¿has solicitado tu vida laboral digitalmente?

- Sí
- No. Necesité personarme en las oficinas o pedir ayuda por otro medio
- No. No he tenido necesidad pero sabría hacerlo
- No conocía esa posibilidad

¿Sabrías consultar digitalmente el distintivo ambiental de tu vehículo?

- Sí
- No
- No conocía esa posibilidad

¿Sabes cómo hacerte socio de tu biblioteca desde casa?

- Sí
- No
- No sé si mi biblioteca tiene esa posibilidad

En los últimos 12 meses, ¿has intentado pedir cita con los servicios sociales de tu localidad de forma digital?

- Sí
- No. Siempre recurro al teléfono o la solicitud presencial
- No. No he tenido necesidad pero sabría hacerlo
- No sé si mi municipio tiene esa posibilidad

Sección 4: Experiencia de usuario

¿Sabes qué organismo es el encargado de concederte la licencia para ocupar la vía pública en una mudanza?

- Policía Nacional
- DGT
- Mi Comunidad Autónoma
- El juzgado
- Mi ayuntamiento
- No necesito licencia
- Otra

Cuando tienes dudas de quién y cómo realizar un trámite con la administración, ¿qué herramientas utilizas?

- Utilizo un buscador generalista (Google, Yahoo, Bing…)
- Llamo al 012
- Recurro a la web de atención al ciudadano de mi comunidad autónoma
- Compruebo la web de mi ayuntamiento

- Me persono en la oficina de atención al ciudadano de mi ayuntamiento
- Otra

En tu experiencia, ¿consideras que la administración ofrece herramientas digitales sencillas?

- Sí, casi todas las administraciones que uso son fáciles de usar
- Sí, pero no para todo el mundo
- No, prefiero dejar de hacer algunas cosas si solo son digitales
- No, siempre necesito otras vías
- Otra

Consideras que una web con los procedimientos comunes de todas administraciones explicados, ¿sería más fácil de manejar?

- Sí
- No
- Tal vez
- Otra

Sección 5: Opinión abierta

Si hay alguna cuestión que te gustaría contarnos respecto al acercamiento a las Administraciones Públicas a través de internet, ¡somos todo oídos!

b. Respuestas (en desarrollo: falta actualizar los datos de la encuesta en este informe para analizar sus resultados en el apartado correspondiente)

Las respuestas que se muestran a continuación son provisionales, ya que se proporcionan con fecha 9 de noviembre de 2022:

Total de respuestas: 50

Sección 2: Perfil de usuario

¿Cuántos años tienes?

- Entre 20 y 29: 16%
- Entre 30 y 44: 48%
- Entre 45 y 59: 24%
- Entre 60 y 76: 12%

¿Con qué género te identificas?

- Femenino: 60%
- Masculino: 40%

¿Desde qué Comunidad Autónoma realizas esta encuesta?

- Andalucía: 12%
- Aragón: 8%
- Castilla y León: 24%
- Cataluña: 20%
- Galicia: 4%
- Madrid: 12%
- País Vasco: 20%

¿Cómo es la población en la que resides actualmente?

- Menos de 10.000 habitantes: 20%

- Entre 10.000 y 25.000: 8%
- Entre 25.000 y 50.000: 16%
- Entre 50.000 y 100.000: 12%
- Entre 100.000 y 250.000: 12%
- Más de 250.000 habitantes: 32%

¿Cuál es tu máximo nivel de estudios alcanzado?
- Estudios primarios (Graduado escolar): 4%
- Secundarios incompletos: 4%
- Bachillerato o Ciclo Formativo: 8%
- Grado universitario: 40%
- Máster universitario y/o Doctorado: 44%

¿Tienes acceso a internet en casa?
- Sí: 96%
- No: 4%

¿A cuál de los siguientes dispositivos electrónicos tienes acceso?
- Teléfono fijo: 60%
- Teléfono móvil sin conexión a internet: 4%
- Smartphone (móvil con conexión a internet con datos o Wi-Fi): 100%
- Tablet: 80%
- Ordenador portátil: 88%
- Ordenador de sobremesa: 48%
- Smart TV (Televisión con conexión a internet): 72%
- Consola de videojuegos con acceso a internet: 44%

¿Utilizas alguna de las siguientes redes sociales?
- WhatsApp: 100%
- Instagram: 76%
- Facebook: 64%
- Tik tok: 28%
- Twitter: 44%
- YouTube: 76%
- Telegram: 48%
- Discord: 28%
- Skype: 4%

Sección 3: Administración electrónica

¿Tienes certificado electrónico o acreditación equivalente digital?
- Sí: 60%
- No: 40%

¿Sabes encontrar el CIP o CIPA de tu tarjeta sanitaria para comprobar tus datos médicos digitalmente?
- Sí, siempre realizo las cuestiones médicas telemáticamente: 56%
- Sí, aunque suelo usar la web o la app solo algunas veces: 36%
- No. Siempre prefiero acudir en persona o llamar por teléfono: 8%

En los últimos 12 meses, ¿has solicitado cita para renovar tu DNI o Pasaporte digitalmente?
- Sí. Aboné las tasas telemáticamente: 20%
- Sí. Aboné las tasas en el momento de realizarlo: 20%
- No. No he tenido necesidad pero sabría hacerlo: 48%
- No conocía esa posibilidad: 12%

En los últimos 12 meses, ¿has solicitado tu vida laboral digitalmente?
- Sí: 56%
- No. Necesité personarme en las oficinas o pedir ayuda por otro medio: 4%
- No. No he tenido necesidad pero sabría hacerlo: 36%
- No conocía esa posibilidad: 4%

¿Sabrías consultar digitalmente el distintivo ambiental de tu vehículo?
- Sí: 52%
- No: 16%
- No conocía esa posibilidad: 32%

¿Sabes cómo hacerte socio de tu biblioteca desde casa?
- Sí: 32%
- No: 12%
- No sé si mi biblioteca tiene esa posibilidad: 56%

En los últimos 12 meses, ¿has intentado pedir cita con los servicios sociales de tu localidad de forma digital?
- Sí: 20%
- No. Siempre recurro al teléfono o la solicitud presencial: 12%
- No. No he tenido necesidad pero sabría hacerlo: 52%
- No sé si mi municipio tiene esa posibilidad: 16%

Sección 4: Experiencia de usuario

¿Sabes qué organismo es el encargado de concederte la licencia para ocupar la vía pública en una mudanza?
- Mi ayuntamiento: 88%
- Otra: 12% ("No", "No lo sé", "¿Se puede pedir licencia para esto?")

Cuando tienes dudas de quién y cómo realizar un trámite con la administración, ¿qué herramientas utilizas?
- Utilizo un buscador generalista (Google, Yahoo, Bing…): 80%
- Llamo al 012: 4%
- Recurro a la web de atención al ciudadano de mi comunidad autónoma: 32%
- Compruebo la web de mi ayuntamiento: 40%
- Me persono en la oficina de atención al ciudadano de mi ayuntamiento: 12%

En tu experiencia, ¿consideras que la administración ofrece herramientas digitales sencillas?
- Sí, casi todas las administraciones que uso son fáciles de usar: 8%
- Sí, pero no para todo el mundo: 44%
- No, prefiero dejar de hacer algunas cosas si solo son digitales: 16%
- No, siempre necesito otras vías: 16%
- Otra: 16% ("No, aunque me suelo apañar", "No creo que sean sencillas, siempre es difícil y frustrante", "Las herramientas de la administración suelen ser engorrosas", "Las herramientas están muy mal diseñadas, las uso y llego a encontrar las cosas. Pero no son para gente mayor o con poco conocimiento de internet")

Consideras que una web con los procedimientos comunes de todas administraciones explicados, ¿sería más fácil de manejar?

- Sí: 84%
- Tal vez: 12%
- Otra: 1% ("Tal vez, pero también un modo tutoría o similar que solo si eres buen conocedor y te sientas cómo con la navegación puedas desactivar")

Sección 5: Opinión abierta

Si hay alguna cuestión que te gustaría contarnos respecto al acercamiento a las Administraciones Públicas a través de internet, ¡somos todo oídos!

- "Cada una funciona diferente, entre comunidades y entre provincias dentro de la misma comunidad"
- "Deberían utilizar un lenguaje más cercano al ciudadano y no tan 'boe'. Hay veces que encontrar lo necesario es complicado"

9.2. Herramientas del proyecto

En el transcurso de esta memoria se ha hablado de muchas herramientas tanto físicas como lógicas para llevar a cabo este trabajo, esta sección anexa intenta aunar todas ellas en un mismo lugar.

a. Hardware

- Ordenador personal[10]: Mac Studio. Apple M1 Max. 64GB de RAM

b. Software

- Visual Studio Code: Versión 1.74.3
- Docker desktop: Versión 4.13.1
- TablePlus: Versión 5.2.2
- Postman: Versión 10.8.1
- Google Chrome Canary: Versión 111.0.5543.0
- Balsamiq Wireframe: Versión 4.6.4
- Affinity Designer: Versión 2.0.3
- QuickTime Player: Versión 10.5
- DaVinci Resolve: Versión 18.1.2
- Keynote: Versión 12.2.1
- Notion: Versión 2.1.11

[10] El ordenador de inicio tuvo que ser reemplazado por el mencionado tras la primera entrega tras un problema técnico

c. Software en línea

- Formularios de Google: https://docs.google.com/forms/create
- Diagrams: https://app.diagrams.net/
- Canva: https://www.canva.com/
- GitHub: https://github.com/
- Heroku: https://id.heroku.com/login
- Netlify: https://www.netlifycms.org/
- Canva: https://www.canva.com/

d. Versiones relevantes

- Node: Versión 18.12.1
- npm: Versión 8.19.2
- Git: Versión 2.37.1
- Angular CLI: Versión 15.0.4
- Nest CLI: Versión 9.1.5
- Heroku CLI: Versión 7.67.1
- Netlify CLI: Versión 12.7.2

9.3. Instrucciones de uso

Para comprobar el funcionamiento de la aplicación en la nube debe accederse a la siguiente dirección y hacerse uso de alguna de las contraseñas facilitadas en un documento aparte para probar su funcionamiento interno.

< https://adfuoc.netlify.app/ >

Para hacer uso de la aplicación en desarrollo es necesario abrir el proyecto del backend con un editor de texto en el que se tenga configurado Node con un gestor de paquetes como npm, yarn o similar e instalar el proyecto con la instrucción:

< npm i >

A continuación, y una vez cargadas instalado el proyecto y las dependencias, abriremos Docker y esperaremos a que inicien sus motores para poder levantar la base de datos mediante la instrucción:

< docker-compose up -d >

Finalmente iniciaremos el proyecto de backend mediante la línea de comando con:

< npm run start:dev >

Una vez que todo el proceso anterior haya concluido con éxito, podemos ejecutar el frontend. En este caso necesitaremos seguir un proceso de instalación del mismo con el mismo comando que usamos para Nest:

< npm i >

Tras instalar satisfactoriamente el proyecto de Angular y sus dependencias podemos echarlo a andar con la instrucción:

< ng serve -o >

Podemos ver el proyecto en funcionamiento en el puerto 4200 de localhost, y en el puerto 3000/api tendremos acceso a la documentación de la API