UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DEL SOFTWARE

Aplicación web de gestión clínica basada en Power Apps

Web application for clinic management based on Power Apps

Realizado por
Miriam Rodríguez Álvarez

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, JUNIO de 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA DEL SOFTWARE

Aplicación web de gestión clínica basada en Power Apps

Web application for clinic management based on Power Apps

Realizado por

Míriam Rodríguez Álvarez

Tutorizado por

Eduardo Guzmán de los Riscos

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2023

Resumen

Las tecnologías low-code/no-code surgen como solución a algunas desventajas de los métodos de programación tradicionales, reduciendo los tiempos y costes de desarrollo del software. La complejidad y el grado de especialidad técnica necesaria para cada tecnología concreta se presenta como un obstáculo para muchas empresas que buscan desarrollar aplicaciones personalizadas orientadas, ya sea bien, a ayudar con procesos internos o satisfacer las necesidades específicas de un cliente. Eliminando la necesidad de tener profundos conocimientos de programación, usando interfaces visuales y herramientas gráficas, estas tecnologías low-code/no-code permiten a los usuarios crear aplicaciones de manera más sencilla, más rápida e igualmente personalizables.

Este Trabajo de Fin de Grado aborda cómo podemos ofrecer una solución a un caso de negocio real, usando la tecnología low-code/no-code que nos ofrece Microsoft, la llamada Power Platform. En concreto el objetivo es crear una aplicación de gestión interna para una clínica de podología local, centrándonos en partes primordiales de cualquier proyecto software como pueden ser: el análisis de requisitos, el diseño del modelo de datos o el diseño de la arquitectura software; siguiendo una metodología de desarrollo ágil.

En el proyecto hacemos uso de Power Apps para la creación de la aplicación principal para navegador y una aplicación con funcionalidades complementarias en tablet, junto con Power Automate para automatizar los procesos lógicos y el procesamiento de los datos y el uso de scripts para una mayor personalización.

Palabras clave:

Low-code/no-code, Power Platform, Power Apps, Power Automate

Abstract

Low-code/no-code technologies emerge as a solution to the drawbacks of traditional programming methods, reducing software development time and costs. The complexity and degree of technical expertise required for a specific technology are viewed as an obstacle for many companies that aim to develop custom applications that facilitate internal processes or fulfill the specific requirements for a client. By eliminating the need for deep programming knowledge and using visual interfaces and graphical tools, these low-code/no-code technologies allow users to create applications more easily, more quickly, and equally customizable.

This Final Year Dissertation addresses how we can offer a solution for a real business case, using the low-code/no-code technology offered by Microsoft, Power Platform. In particular, we will create an internal management application for a local podiatry clinic, focusing on primary parts of any software development project such as requirements analysis, data model design, or software architecture design, following an agile methodology.

In the project, we make use of Power Apps to create the main browser-based application and a complementary tablet application, along with Power Automate to automate logical processes and data processing, and the use of scripts for further customization.

Keywords:

Low-code/no-code, Power Platform, Power Apps, Power Automate

Índice

1. Introducción

1.1 Motivación

En los últimos años, la industria del software ha sido testigo de un cambio hacia el uso de tecnologías low-code/no-code en el desarrollo de aplicaciones. Esto se debe a la creciente demanda de soluciones rápidas y eficientes, que permitan a las empresas reducir los tiempos y costes asociados al desarrollo de software. La utilización de interfaces visuales y herramientas gráficas ha permitido a los usuarios crear aplicaciones personalizadas con facilidad, sin la necesidad de tener amplios conocimientos de programación. Esta tendencia ha sido impulsada por el aumento en la cantidad de proveedores de tecnologías low-code/no-code en el mercado, y por la aceptación cada vez mayor de estas soluciones por parte de la industria.

Ante esta situación, la tecnología Power Platform de Microsoft se ha destacado como una de las soluciones líderes en el mercado de tecnologías low-code/no-code. Al hacer uso de Power Apps, Power Automate, Power BI y Power Agents, los usuarios pueden crear aplicaciones personalizadas y soluciones empresariales eficientes y escalables.

1.2 Objetivos

En este Trabajo de Fin de Grado, se ha propuesto el uso de Power Platform para desarrollar una aplicación de gestión interna para una clínica de podología local, que permita mejorar la eficiencia de sus procesos y aumentar su productividad. Esta aplicación se centrará en partes primordiales de cualquier proyecto de software, como el análisis de requisitos, el diseño del modelo de datos y la arquitectura del software, siguiendo una metodología de desarrollo ágil.

La metodología escogida para el desarrollo del proyecto será ágil e iterativa, planeando 4 iteraciones con una fase inicial de análisis en cada iteración y una fase final donde se comentan las conclusiones de cada iteración con respecto al desarrollo de la siguiente. Además, se realizará un análisis inicial de la aplicación como punto de partida.

En la primera iteración, se implementará el modelo de datos, los primeros formularios y vistas de la aplicación. En la segunda iteración, se llevará a cabo el desarrollo de los procesos automáticos que soportan la lógica de la base de la aplicación. En la tercera iteración, se mejorarán los procesos automáticos y las interfaces de la aplicación. Finalmente, en la cuarta y última iteración, se llevará a cabo la integración con aplicaciones fuera de la aplicación base o con otros servicios y procesos que permitan añadir funcionalidades más complejas.

Como resultado final del proyecto, se generará como entregable un archivo comprimido con los componentes de la aplicación, el cual permitirá su despliegue en otros entornos si fuera necesario.

1.3 Estructura de la memoria

Esta memoria está organizada en 5 secciones, que describen todo el progreso en la realización de este Trabajo de Fin de Grado. A continuación, se presenta una breve introducción a cada una de ellas.

- El capítulo 1, “Introducción”, establece la motivación y los objetivos del trabajo, describiendo la base para el proyecto que se ha desarrollado.
- El capítulo 2, “Conceptos y tecnologías”, explora las tecnologías “Low-code/no-code” y la plataforma Power Platform, justificando su elección por encima del resto de tecnologías.
- El capítulo 3, “Análisis e iniciación del proyecto”, analiza el caso de negocio, estudia aplicaciones o servicios similares, las licencias de Power Apps y presenta el diseño a alto nivel de la solución. Después, desarrolla el proceso

de iniciación del proyecto, explicando el uso de entornos y soluciones, creación de nuevos editores y prefijos, etc.

- El capítulo 4, “Desarrollo del proyecto”, se detalla el proceso de desarrollo de la aplicación, gestionada por sprints o diferentes iteraciones, incluyendo los requisitos y la resolución de cada uno.
- Finalmente, el capítulo 5, “Conclusiones y líneas futuras”, está dedicado a las conclusiones a las que se ha llegado como consecuencia de la realización de la aplicación, así como las posibles direcciones para continuar su desarrollo en un futuro.

De manera adicional, se incluye como apéndice un “Manual de Instalación” para proporcionar una guía paso a paso sobre cómo instalar la aplicación.

2. Conceptos y tecnologías

En esta sección, introduciremos el concepto de las plataformas de desarrollo “low-code/no-code”, que cada vez se están convirtiendo en elecciones más populares en el desarrollo de software, especialmente gracias a la rapidez para desarrollar pequeñas aplicaciones y su fácil despliegue. Además, se dará una visión introductoria sobre la tecnología seleccionada, Power Platform.

2.1 Tecnologías “Low-code/no-code”

Las plataformas low-code/no-code son entornos de desarrollo que permiten la creación de aplicaciones software con un mínimo esfuerzo. Estas plataformas ofrecen para el desarrollador una serie de interfaces visuales y componentes pre-construidos, que permiten desarrollar, realizar pruebas y desplegar aplicaciones rápidamente. A continuación vamos a tratar las ventajas y desventajas que tienen con respecto a las tecnologías tradicionales. [1]

Algunas de las ventajas de estas plataformas son las siguientes:

- Desarrollo y despliegue más rápidos. Gracias a los componentes pre-construidos e interfaces visuales, los desarrolladores pueden crear aplicaciones reduciendo el tiempo de codificación y su despliegue.
- Mayor accesibilidad con respecto al nivel de conocimiento técnico y habilidades de programación especializadas, lo cual reduce los costos y aporta facilidades a los usuarios no técnicos de cara a su participación en el desarrollo.
- Mantenimiento y colaboración más fáciles, ya que estas plataformas de desarrollo suelen proporcionar herramientas centralizadas para administrar y actualizar aplicaciones.

- Integración, ya incorporada, con otros sistemas, lo que permite que las aplicaciones se conecten con varias fuentes de datos y servicios.

Sin embargo, a pesar de proporcionar numerosos beneficios, existen ciertas desventajas que condicionan tanto el proceso de desarrollo como el producto final:

- La personalización y la flexibilidad son limitadas. Si bien estas plataformas ofrecen diversos componentes y plantillas, estos pueden no cubrir todos los casos de uso posibles o adaptarse a requisitos más específicos, lo que dificulta la implementación de lógicas complejas.
- Se crea una dependencia con el proveedor de la plataforma. Al desarrollarse la aplicación en el ecosistema del proveedor con sus herramientas y facilidades, migrarlas a otras plataformas o integrarlas con servicios de terceros puede generar dificultades.
- Existen ciertos compromisos de rendimiento dependiendo de la plataforma. Al estar enfocadas normalmente para el desarrollo de aplicaciones de menor escala y su rápido despliegue, podrían no estar optimizadas para trabajar con grandes conjuntos de datos u operaciones complejas. Un estudio previo de la plataforma es necesario para evitar este tipo de problemas.
- La curva de aprendizaje puede ser elevada al principio, ya que a pesar de que las plataformas están diseñadas para ser sencillas de utilizar, para aquellos usuarios que no estén familiarizados con este tipo de herramientas puede resultar más complicado debido a la cantidad de herramientas disponibles.

Algunas plataformas populares son Microsoft Power Platform, plataforma low-code que proporciona una colección de herramientas del ecosistema de Microsoft que permiten automatizar flujos de trabajo, visualizar datos y crear aplicaciones customizadas; OutSystems [2], plataforma low-code que proporciona plantillas para el desarrollo de aplicaciones web y móviles; Bubble [3], una plataforma no-code que

permite a los desarrolladores crear aplicaciones web utilizando una interfaz drag-and-drop, empleada principalmente para la creación de prototipos; etc.

2.2 El ecosistema de aplicaciones Power Platform

En el desarrollo de la aplicación presentada en este trabajo, se ha optado por utilizar Microsoft Power Platform como plataforma principal. Esta decisión se fundamenta en la sólida reputación y versatilidad de Power Platform, que se ha consolidado como una solución PaaS ampliamente utilizada y reconocida entre los desarrolladores.

A continuación definimos una serie de ventajas y de funcionalidades:

Una de las ventajas clave de Microsoft Power Platform es la presencia de una comunidad activa y dinámica de desarrolladores. Esto facilita el acceso a una gran cantidad de recursos, conocimientos y soluciones, proporcionando un apoyo invaluable en la resolución de problemas y en la búsqueda de mejores prácticas. La comunidad también permite el intercambio de experiencias y conocimientos sobre escenarios y requisitos previamente identificados por otros usuarios, lo que enriquece el proceso de desarrollo y mejora la calidad del producto final. [4]

Microsoft Power Platform es un proyecto maduro que ofrece una amplia gama de funcionalidades, lo que aumenta su adaptabilidad en diversos escenarios y aplicaciones. En particular, para la creación de componentes personalizados que lleven a cabo funcionalidades específicas, la plataforma permite la inclusión de código en el lenguaje Power Fx. Este lenguaje funcional, de sintaxis similar a Excel, permite expresar lógica de negocio, manipular datos, controlar el flujo de la aplicación y personalizar la apariencia de los componentes de la interfaz de usuario.

Además, permite interactuar con APIs y bases de datos externas, permitiendo la integración de fuentes externas en la aplicación. Esto proporciona una mayor flexibilidad para adaptar las plantillas y realizar acciones concretas según las necesidades del caso de estudio. [5]

Otro aspecto importante es su modelo de coste por uso. Existen diferentes modelos de licencias que se adaptan al tamaño y las necesidades de tu empresa, esto la convierte en una opción económica y escalable [6].

En cuanto a la seguridad, Microsoft Power Platform cuenta con funciones integradas que permiten controlar el acceso a los datos y funcionalidades. Además, se integra con Microsoft 365 y Azure Active Directory, [7], permitiendo una gestión unificada de identidades y accesos.

Un aspecto fundamental en el desarrollo de aplicaciones es la utilización de entornos y soluciones. Los entornos [8] son espacios aislados donde se almacenan, gestionan y comparten recursos, como datos, aplicaciones y flujos de trabajo. Cada entorno se puede configurar de manera distinta según su objetivo, como por ejemplo, entornos de producción como productos finales, o de desarrollo para la realización de pruebas e implementación de nuevas funcionalidades. Por otro lado, las soluciones [9] se encargan de empaquetar y administrar las aplicaciones y sus componentes. Esto permite exportar soluciones y gestionar dependencias entre componentes. El disponer de ambos elementos proporciona un marco robusto para el desarrollo, prueba, despliegue y mantenimiento de aplicaciones, permitiendo una colaboración efectiva y una gestión eficiente del ciclo de vida de la aplicación.

En resumen, Microsoft Power Platform ha sido seleccionada para el desarrollo de la aplicación en este trabajo de investigación debido a su capacidad, reputación, comunidad activa y funcionalidades avanzadas. Estos factores, en conjunto, ofrecen una base sólida y adaptable para llevar a cabo el proyecto con éxito y satisfacer las necesidades específicas del caso de estudio en cuestión.

3. Análisis e iniciación del proyecto

En esta sección se va a tratar todos los elementos que hay que tener en cuenta antes de empezar con el desarrollo de la aplicación, definiendo así los procesos de análisis que se requieren para decidir el uso de la tecnología hasta los diagramas que se usarán para dar una descripción más concreta del sistema que disponemos a construir. También se incluyen algunas guías sobre los pasos iniciales para comenzar a trabajar con la tecnología, inicializar el proyecto y algunas directrices básicas que serán tomadas en cuenta durante el desarrollo y el análisis de los requisitos.

Para construir un sistema de software exitoso y efectivo, es fundamental contar con un análisis exhaustivo y riguroso de los requisitos del sistema y del entorno en el que se implementará. En este sentido, uno de los primeros procesos que se deben definir son los análisis que se llevarán a cabo para evaluar y determinar el uso adecuado de la tecnología, teniendo en cuenta las necesidades específicas de la aplicación, así como las limitaciones técnicas y financieras que puedan existir.

Además, otro elemento fundamental que se aborda en esta sección es la definición de los diagramas que se utilizarán para proporcionar una descripción más concreta y detallada del sistema que se pretende construir, como por ejemplo el modelo de datos o el diagrama de arquitectura software.

En última instancia, esta sección del proceso de desarrollo de software es clave para garantizar que la aplicación se construya de manera eficiente y efectiva, tomando en cuenta todos los elementos necesarios antes de comenzar con la codificación y la implementación. Este análisis previo es esencial ya que su objetivo

es evitar problemas y errores en etapas posteriores del proceso de desarrollo, mejorando la calidad, el rendimiento y la eficacia de la aplicación final.

3.1 Análisis previo

La sección 'Análisis previo' sienta las bases para nuestro proyecto, desglosando varios aspectos clave que nos guiarán durante todo el proceso de desarrollo. Nos enfocaremos en entender el contexto del negocio, las herramientas disponibles y cómo planeamos utilizarlas. Analizaremos el diseño general de la solución, cómo se estructurará la información y cómo se manejarán los datos en la plataforma.

3.1.1 Caso de negocio

Este proyecto responde a la necesidad específica de una clínica de podología. Buscan una solución digital que agilice y organice sus actividades diarias, centrándose en el paciente y optimizando el tiempo de los profesionales de la clínica. Inicialmente, la aplicación sería utilizada por el propietario de la clínica, pero está diseñada para ser escalable y adaptarse al crecimiento futuro, permitiendo la incorporación de más doctores.

La clínica requiere una aplicación que almacene de manera segura y accesible datos críticos del paciente, así como de las citas. Un aspecto clave de este almacenamiento de datos es la capacidad para hacer seguimiento de los pacientes, incluyendo un historial de citas, los tratamientos administrados y los diagnósticos emitidos. Para ciertos procedimientos, como los que utilizan láser, es necesario obtener un consentimiento firmado del paciente. Por lo tanto, la aplicación necesitará una funcionalidad para almacenar estas firmas de consentimiento digitalmente.

La aplicación estará basada en una Model Driven App con Dataverse como base de datos. Para optimizar la funcionalidad y la eficiencia, se integrará con Cloud Flows (Power Automate) y una Canvas App. La Canvas App permitirá recoger las firmas de

los pacientes de manera digital, eliminando la necesidad de escanear documentos en papel y agilizando el proceso.

En el aspecto de la gestión de citas, la aplicación enviará recordatorios automáticos a los pacientes, ya sea por SMS o por correo electrónico, un día antes de su cita. Para optimizar aún más el proceso a la hora de gestionar los datos de citas, el propio sistema ofrecerá varias automatizaciones, autocompletando datos, para que sea más fácil para el usuario registrar las citas.

Durante el desarrollo del proyecto, se investigarán aplicaciones similares como Clinic Cloud, ClinicApp y SmartClinicWeb para obtener ideas y mejores prácticas y también se tendrán en cuenta los requisitos de protección de datos [10].

3.1.2. Aplicaciones similares

Se llevó a cabo un análisis de las aplicaciones web que ofrecen soluciones comparables a las propuestas en este estudio. A continuación, se proporciona una breve descripción de cada una, subrayando sus características y sus costos.

ClinicCloud, [11], es un servicio de administración de clínicas basado en la nube que se enfoca en la optimización y el control de los recursos clínicos. Esta plataforma ofrece la posibilidad de establecer perfiles para los profesionales, cada uno con su propia agenda personalizada, gestionar fichas de pacientes, llevar un seguimiento de facturas, notificaciones y alertas mediante correo electrónico y SMS, entre otras cosas. Además, la plataforma se extiende hasta la gestión estratégica del negocio, realizando un seguimiento automatizado de las fluctuaciones mensuales de ingresos y gastos, permitiendo la creación de planes de pago con cuotas recurrentes y gestionando el stock de productos. Su estructura de precios es variada, con el plan Mini, de 32€ al mes, que tiene algunas limitaciones, o el plan Pro, de 57€ al mes, con todas las características y la opción de personalizar el plan según las necesidades.

El software en la nube ClinicApp, [12], es una solución basada en la nube que se centra en la gestión de citas y registros médicos. La plataforma ofrece la posibilidad de programar citas individuales o grupales, organizar citas online, mantener agendas personalizadas y administrar el histórico de sesiones. Ofrece una serie de planes de suscripción que se adaptan a las necesidades del usuario, desde una versión más simplificada para un solo usuario, de 9.90€ al mes, hasta opciones empresariales por 39.90€ al mes y soluciones personalizadas de un total de 490€ al mes.

Otra solución en la nube es SmartClinicWeb, [13], que proporciona una solución integral para la administración de clínicas médicas y la gestión de pacientes. Ofrece las mismas funcionalidades que ClinicApp, pero añade capacidades para gestionar múltiples centros y generar estadísticas sobre las transacciones económicas entre la clínica y los pacientes. El software va más allá de la simple gestión de citas y pacientes, ofreciendo funcionalidades para manejar toda la estructura de negocio. Su precio es de 39.95€ al mes por centro médico.

Tanto ClinicApp como SmartClinicWeb cuentan con un periodo de prueba gratuito de 15 días y 1 mes respectivamente, y todas ellas requieren contactar con ellos para utilizar y solicitar el software a medida en base a tus requisitos.

3.1.3 Licencias de power apps

Microsoft ofrece varias opciones de licencias para Power Apps que se adaptan a diferentes necesidades y presupuestos. En un caso de negocio real es importante considerar estas opciones cuidadosamente, ya que pueden limitar las capacidades del sistema [6].

- Licencia por usuario: Esta licencia permite a un usuario individual crear y ejecutar aplicaciones ilimitadas, incluyendo aplicaciones de Canvas y

Model-driven, así como Portales de Power Apps. Su coste mensual es de 18.70€.

- Licencia por aplicación: Esta opción permite a un usuario individual ejecutar una aplicación específica. Es ideal para situaciones en las que los usuarios necesitan una o dos aplicaciones para realizar su trabajo. El coste por usuario y por aplicación es de 4.70€/mes,
- Plan de pago por uso: Esta licencia utiliza una suscripción de Azure para pagar por usuario en función del número de aplicaciones únicas que un usuario ejecuta cada mes. Los costos varían en función del número de visitas de página que se espera recibir y del cómputo realizado.

Es importante tener en cuenta que cada una de estas licencias tiene diferentes limitaciones en términos de desarrollo. Por ejemplo, la licencia por usuario permite un desarrollo más extenso y personalizado, ya que da acceso a un número ilimitado de aplicaciones. Por otro lado, la licencia por aplicación puede ser más limitada en términos de desarrollo, ya que está vinculada a una o dos aplicaciones específicas.
Además, es importante mencionar que las licencias de Power Apps también incluyen acceso a Dataverse, la plataforma de datos de Microsoft. Sin embargo, el uso de Dataverse puede verse influido según el tipo de licencia de Power Apps o Power Automate que se disponga.

3.1.4 Diseño a alto nivel de la solución

El diagrama de la Figura 1 proporciona una visión a alto nivel del diseño de la aplicación y de cómo se integran sus diversos componentes. En la primera columna, "Usuarios finales", se sitúa el punto de partida del flujo de trabajo de la aplicación, mostrando que los usuarios interactúan con la aplicación a través de diferentes dispositivos, como navegadores web, tablets y móviles. En la segunda columna, "Aplicaciones Power Apps", se describen las aplicaciones Canvas y Model Driven que se utilizan en la aplicación.

La tercera columna, "Fuente de datos", destaca el papel central que desempeña Dataverse como el almacén de datos principal de la aplicación, que alimenta y recibe información de las aplicaciones y procesos que componen el sistema. Finalmente, en la cuarta columna, "Procesos automatizados Power Automate", se enumeran los distintos flujos que se utilizan para automatizar procesos, agilizar el flujo de trabajo y mejorar la eficiencia en toda la aplicación.

Este diagrama hace énfasis en la dependencia mutua de estos componentes dentro de la plataforma de Power Platform, y cómo estos elementos trabajan juntos para crear una solución integral que satisfaga las necesidades específicas de la aplicación.

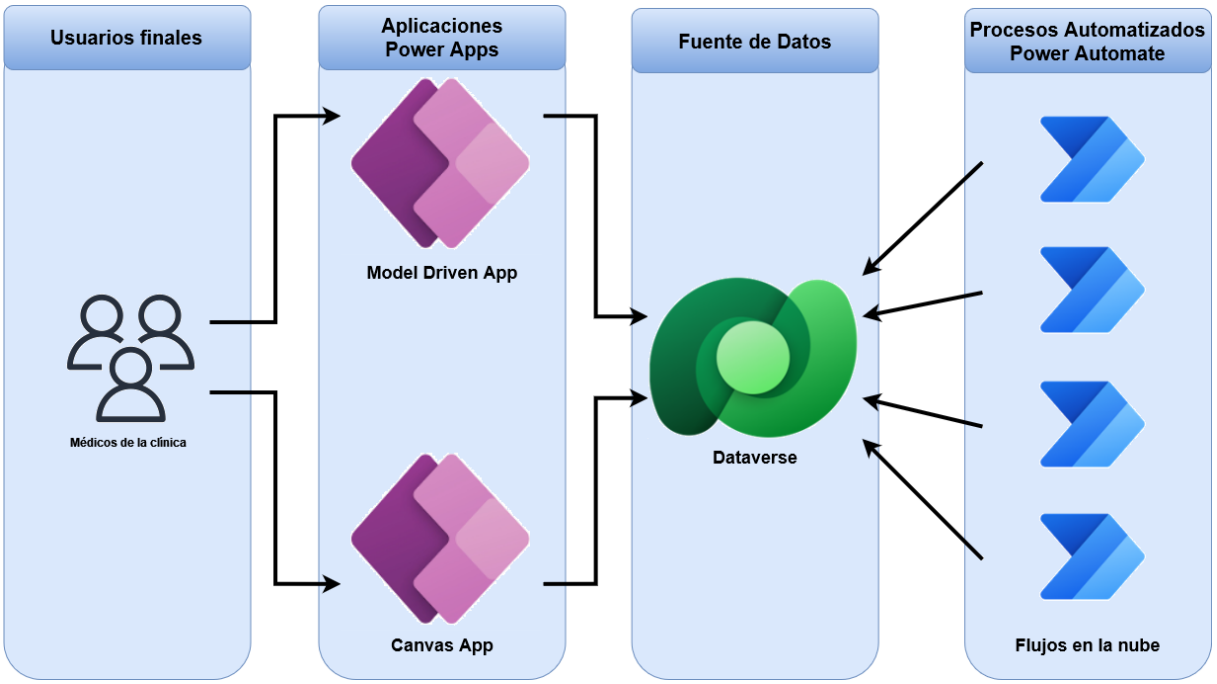

Figura 1. Diagrama de Arquitectura

3.1.5 Estructuras de datos “Out Of the Box” en Dataverse (OOB Entities)

Dataverse, la plataforma de datos de Microsoft, proporciona un conjunto de entidades predefinidas o "Out of the Box" (OOB) que facilitan la creación y el manejo de aplicaciones. Estas entidades OOB son estructuras de datos

prefabricadas que representan objetos comunes del mundo real, como "contactos", "cuentas" o "citas", y son altamente personalizables para adaptarse a las necesidades específicas de una organización o de una aplicación. [14]

Las entidades OOB de Dataverse ofrecen varias ventajas. En primer lugar, permiten un desarrollo más rápido y eficiente, ya que eliminan la necesidad de crear desde cero estructuras de datos para objetos comunes. Además, estas entidades ya están optimizadas para su uso con otras aplicaciones de Microsoft, como Power Apps, Power Automate y Power BI, permitiendo una integración fluida con estas herramientas.

Otra ventaja de las entidades OOB es que vienen con una serie de características predefinidas, como formularios, vistas y campos, que se pueden utilizar tal cual o personalizar según las necesidades. Estas características predefinidas pueden ahorrar una cantidad significativa de tiempo de desarrollo y asegurar la consistencia en la forma en que se manejan ciertos tipos de datos en la aplicación.

En el caso de nuestra aplicación para la clínica de podología, planeamos utilizar varias entidades OOB de Dataverse. Por ejemplo, la entidad "contactos" se puede utilizar para gestionar los datos de los pacientes, mientras que la entidad "citas" se puede utilizar para gestionar las citas de los pacientes. La entidad "cuentas" podría representar a los doctores o al personal de la clínica. Por último, podríamos utilizar la entidad "casos" para manejar el seguimiento de los pacientes, incluyendo su historial de citas, tratamientos y diagnósticos. Todas estas entidades se modificarán o se añadirán nuevos elementos según las necesidades específicas de la clínica de podología.

3.1.6 Modelo de datos

El propósito principal de un modelo de datos en el desarrollo de software es proporcionar una representación visual y estructurada de los datos que el software gestionará y procesará. Desde ahí podremos identificar los requisitos de datos de nuestro software, tales como el tipo de los datos que se necesitan almacenar y las relaciones entre estos.

La Figura 2 presenta una aproximación inicial al modelo de datos que utilizaremos, desde el cual haremos modificaciones hasta llegar al resultado final.

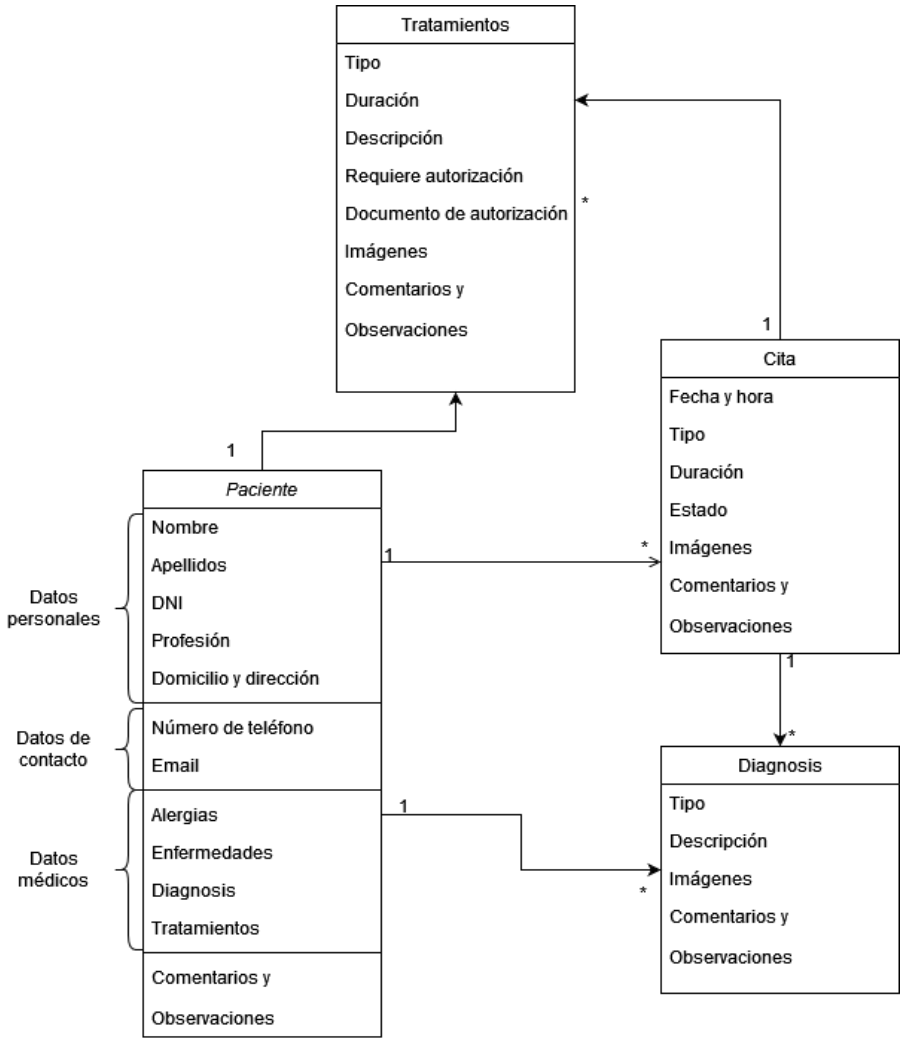

Figura 2: Modelo de datos.

3.2 Iniciación del proyecto

Tras haber estudiado a fondo el caso de negocio, las herramientas disponibles y cómo se planea usarlas, nos disponemos a explicar cómo se iniciará el proyecto. En esta sección, se establecerá la base del proyecto, incluyendo la configuración inicial de la aplicación, convenciones a seguir y una visión general de cómo se distribuyen los componentes en el entorno. Viendo cómo podemos inicializar esos componentes, las acciones requeridas sobre estos y la correcta gestión de las soluciones.

3.2.1 Convenciones de nomenclatura

Antes de empezar a crear componentes o elementos, primero tendremos en cuenta las guías de nomenclatura ofrecidas en la documentación de Microsoft [15].

Algunas de las guías más interesantes son:

- Dos objetos del mismo tipo no deben tener el mismo nombre.
- Evitar los nombres generalistas que puedan llevar a conclusión, los componentes deben tener un nombre que indique claramente su aplicación.
- El nombre principal de las tablas debe ser siempre singular y no en plural.
- A la hora de crear campos, si el campo es de tipo fecha, incluir la palabra “fecha” en el nombre del campo.
- Usar afirmaciones positivas para definir campos de tipo lógico.
- Usar el verbo de la acción principal asociada a un proceso dentro del nombre del mismo.
- Cuando haya algún conflicto entre entidades y se dispongan de dos iguales usar el prefijo “antiguo” para los campos de la entidad antigua y “nuevo” para los campos de la entidad nueva.

La última guía también puede ser interesante cuando necesitamos recrear un campo y no podemos borrar el antiguo campo antes de crear el nuevo, podemos usar el prefijo “nuevo” para indicar cuál de los dos campos es más reciente,

evitando así confundir a los usuarios o a los otros desarrolladores teniendo que usar un nombre distinto y no descriptivo para el nuevo campo que sustituirá al antiguo.

Aplicaremos esta serie de convenciones para facilitar el desarrollo y seguir buenas prácticas, generando así una aplicación con una lógica consistente, simplificando su uso y mantenimiento.

3.2.2 Crear la aplicación

Una vez hemos conseguido acceso a Power Apps, ya sea comprando una licencia o como en el caso de este proyecto, usando una de las licencias gratuitas o "trials" que ofrece Microsoft, nos dispondremos a configurar nuestro entorno [16].

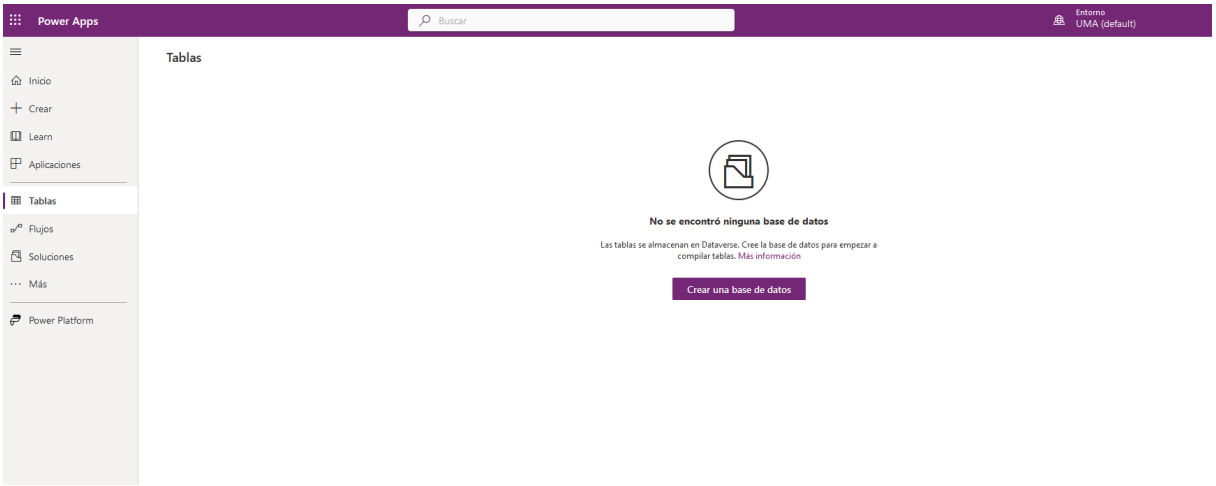

Figura 3. Entorno creado

Tal como vemos en la Figura 3, nada más crear el entorno lo primero que tenemos que hacer es desarrollar nuestra base de datos de Dataverse, la cual se podrá construir con datos de ejemplo para las entidades ya incluidas, la entidades OOB. Además de eso tendremos que decidir varios parámetros en su creación, como observamos en la Figura 4.

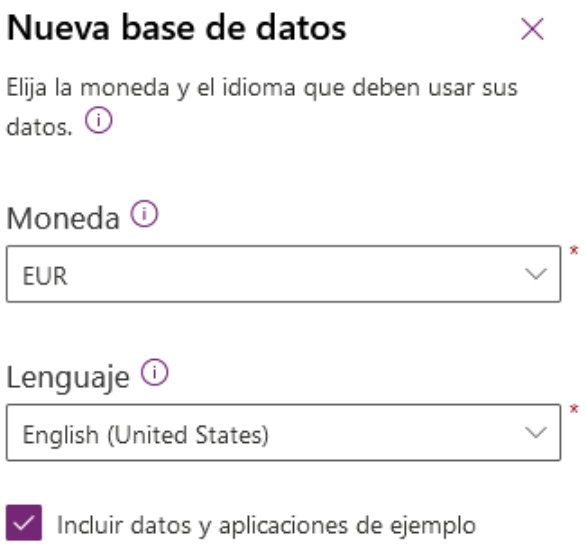

Figura 4. Creación de la nueva base de datos

Cuando el sistema nos avise que la base de datos ya está lista, nuestro siguiente paso será crear la solución que contendrá los componentes con los que construiremos la aplicación tanto como la aplicación en sí misma. Si nos dirigimos en el menú lateral al apartado “Soluciones” podremos seleccionar la opción nueva solución en el menú de arriba.

Figura 5. Creación de una nueva solución

Como podemos ver en la Figura 5, a la hora de crear la solución tenemos que rellenar varios campos, entre ellos vamos a centrarnos en el campo editor o publicador [17]. El editor decidirá el prefijo que tendrán todos los componentes que creemos para esta nueva solución, por lo cual en vez de utilizar el editor por defecto, que ya existe, vamos a crear uno nuevo antes de continuar con la creación de la solución. Nos desviaremos a la pantalla de publicadores o pulsaremos en el botón “nuevo editor” que se ve en la imagen de arriba, para crear nuestro nuevo editor o publicador, una vez listo volveremos a la creación de la solución.

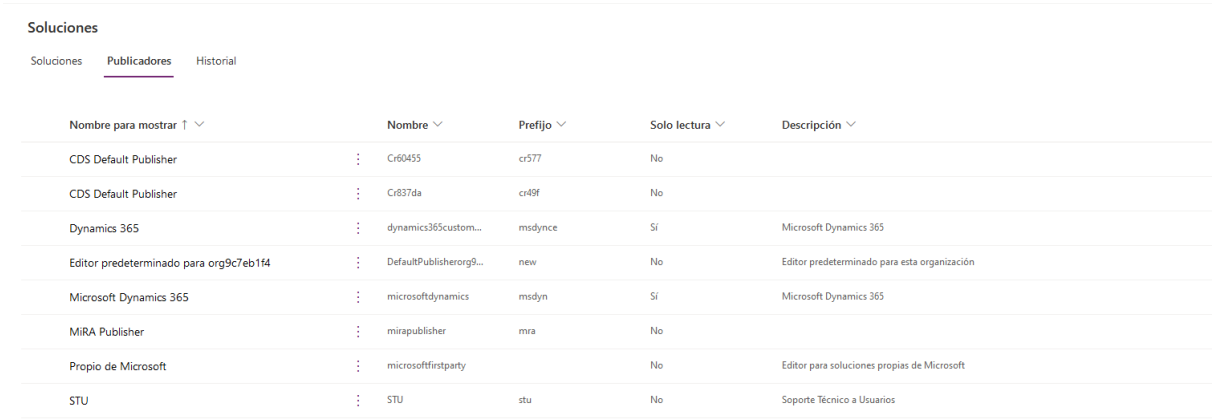

Figura 6. Pantalla de publicadores

En la Figura 6, podemos ver como nuestro nuevo publicador se llama “MiRA Publisher” y añadirá el prefijo “mra” a todos los componentes que creemos al usarlo como editor de la solución. Una vez creado podemos volver a la creación de nuestra solución y seleccionarlo dentro del formulario, rellenaremos todos los campos y llamaremos a la solución “Gestion Clinic App”.

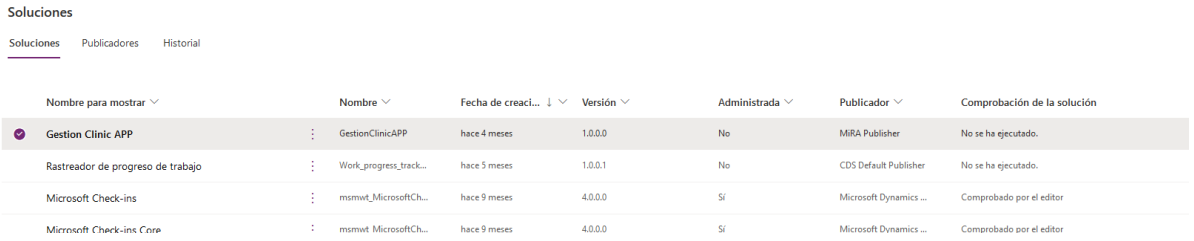

Figura 7. Creación de la solución

Ya con nuestra solución creada, tal y como se observa en la Figura 7, podemos disponernos a crear nuestra model-driven app. Primero entramos dentro de la solución y seleccionamos el botón “Nuevo” del menú superior; entonces se desplegará un menú de opciones con los componentes de uso más frecuente en Power Apps. De todos esos componentes, seleccionaremos el componente de “Aplicación” y desde ahí podemos elegir “Aplicación basada en modelo”. Llamaremos a la aplicación igual que a la solución, “Gestion Clinic App”.

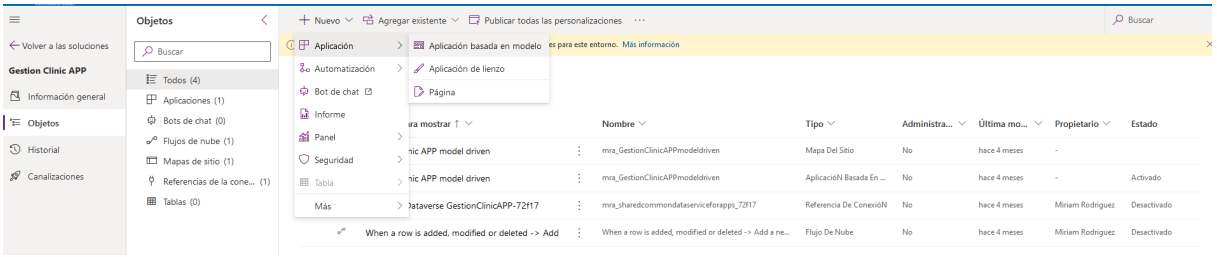

Figura 8. Creación de la aplicación

Con respecto a la solución, podemos observar varias cosas, fijándonos en la Figura 8. Por una parte vemos cómo se organizan los componentes en una solución desde el menú lateral de la solución. Si bien podemos elegir ver todos los componentes juntos, disponemos de varias secciones distintas que nos ayudarán a agrupar los componentes en grupos. Los dos grupos que más llegaremos a usar durante el desarrollo serán los “Flujos de nube” y la sección de “Tablas”.

Entre los componentes que crearemos encontraremos sobre todo las tablas y sus campos, junto con los formularios, vistas y reglas de negocio de cada una de estas tablas. De nuevo, todos estos componentes se encontrarán localizados en la sección “Tablas” de la solución, teniendo cada uno a su vez su propia sección dentro de la sección principal como vemos en la Figura 9. Aparte de estos componentes comunes, también dispondremos de una variedad de flujos en la nube. Por ello estas dos secciones serán de las más utilizadas durante el proyecto, aunque inicialmente también nos centraremos en editar el componente de la aplicación para vincular todos los datos que necesitemos con la misma.

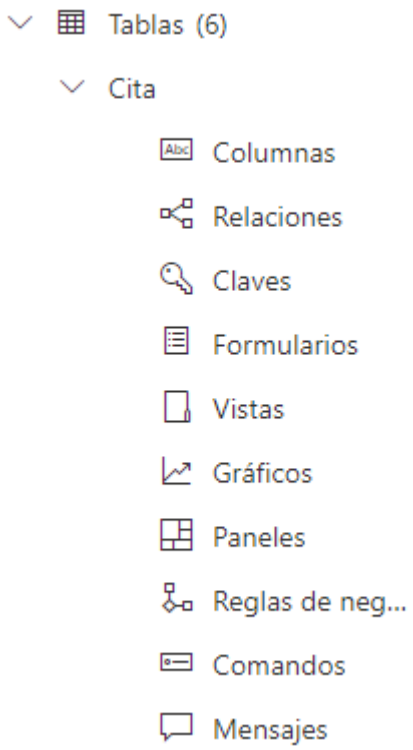

Figura 9. Secciones del menú lateral “Tablas”

Finalmente, con el entorno creado y configurado, la base de datos funcionando con las entidades OOB y los datos de prueba, nuestra solución, que almacenará los componentes del sistema, el editor de la solución, para diferenciar los componentes que creamos usando nuestro prefijo, y la aplicación ya creada, podemos proceder con el inicio del desarrollo, descripción de requisitos y creación de los primeros datos y entidades.

4. Desarrollo del proyecto

4.1 Plantilla de requisitos

Disponer de una tabla o plantilla de especificación de requisitos es una herramienta importante para definir los requerimientos de un proyecto. En concreto esta tabla, está conformada por tres elementos fundamentales que en su conjunto nos permiten describir de forma ordenada la información necesaria para un correcto desarrollo y son esenciales para lograr el éxito del proyecto, estos son: escenario, criterios de aceptación y diseño técnico y componentes de la solución.

Nombre	Título/Identificador del requisito
Escenario	**Como** [tipo de usuario / nombre de la aplicación] -> quién **quiero** [necesidad / acción] -> qué **para** [beneficio esperado / resultado] -> para qué
Descripción	Breve descripción del requisito
Criterios de Aceptación	**Dado que** [Contexto / Prerrequisito] **y adicionalmente** [Contexto / Prerrequisito] (opcionalmente) **cuando** [Evento / Acción] **entonces** [Resultado / Comportamiento esperado]
Diseño técnico	Descripción técnica de cómo resolver el problema
Componentes de la solución	Nombre exacto de los componentes que intervienen en la solución

En primer lugar, se encuentra el "Escenario", donde se describe el contexto y las circunstancias en las que se aplicará el requisito. Aquí se identifica al usuario o la aplicación involucrada, se especifica la necesidad o acción requerida y se describe el beneficio esperado, [18] [19].

En segundo lugar, se encuentran los "Criterios de Aceptación", que establecen las condiciones que deben cumplirse para considerar que el requisito ha sido satisfecho. Estos criterios son necesarios para evitar confusiones y garantizar que todos los involucrados tengan una comprensión clara de lo que se espera [20].

Finalmente, está el "Diseño Técnico", que describe la solución técnica que se utilizará para resolver el problema, usando este diseño somos capaces de identificar los componentes necesarios para la implementación. Esta sección es crucial para asegurar que la solución sea viable y se pueda implementar de manera efectiva.

Ahora que hemos establecido la definición de los elementos principales de nuestra plantilla de especificación de requisitos, es importante entender cómo vamos a utilizarla para el resto del proyecto, ya que esta es una herramienta vital para la planificación, implementación y verificación de todas las funcionalidades propuestas para nuestra aplicación. A la hora de planificar, un documento general de requisitos o backlog [21] sirve como una guía para asegurarse de que el proyecto esté en línea con los objetivos específicos establecidos, nos ayuda a identificar cualquier posible riesgo y a asegurarnos de que se hayan tomado en cuenta todas las variables necesarias.

En nuestro caso concreto vamos a definir un conjunto inicial de requisitos y los vamos a repartir separados en iteraciones o sprints [22] que son ciclos de trabajo usualmente con una duración de unas pocas semanas tras el cual se acaba generando una versión funcional del producto que implementa todas las funciones definidas para el sprint concreto.

En la siguientes secciones se va a describir, usando la plantilla de requisitos que hemos definido, las distintas funcionalidades o requisitos de nuestra aplicación

repartido en un total de cuatro sprints o iteraciones de tal manera que cada sprint tendrá unos objetivos concretos que serán evaluados y presentados al final de los mismos en la sección de conclusiones, pudiendo además incluir, eliminar o modificar requisitos a lo largo del proyecto si fuera necesario, adaptándonos así al avance del desarrollo.

4.2 Desarrollo del primer sprint

Según lo que hemos comentado anteriormente, este sprint o iteración se centrará en el primer acercamiento e implementación del modelo de datos, definiendo cómo serán los datos en su estructura, su tipo y las relaciones entre entidades. Centrándonos sobre todo en las buenas prácticas, tanto para la implementación de los campos como para el uso de los datos ya existentes en Dataverse.

4.2.1 Requisitos para el primer sprint

Nombre	S1.001 - Almacenar datos sobre el paciente
Escenario	**Como** aplicación de gestión clínica **quiero** definir una nueva entidad **para** poder almacenar datos sobre pacientes
Descripción	La aplicación debe ser capaz de contener los datos de los pacientes de la clínica, para ello hay que identificar los datos que es necesario almacenar y el tipo de datos. Posteriormente también deberá considerarse la relación entre la información de pacientes y otra información necesaria para el correcto funcionamiento de la aplicación.

Criterios de Aceptación	**Dado que** se necesita almacenar datos sobre los pacientes **cuando** llegue un paciente **entonces** voy a poder almacenar todos sus datos
Diseño técnico	Power apps dispone de un servicio de base de datos llamado Dataverse, usando Dataverse podemos almacenar todos los datos de pacientes para posteriormente usarlos en nuestra aplicación. Un paciente tendrá como mínimo los siguientes datos: ● Datos Personales ○ Nombre ○ Apellidos ○ DNI ○ Profesión ○ Domicilio y dirección ● Datos de Contacto ○ Número de telefono ○ Email ● Datos Médicos ○ Alergias ○ Enfermedades ○ Diagnósticos ○ Tratamientos ● Comentarios y Observaciones Evaluar si es necesario crear una nueva entidad, de lo contrario, usar la entidad Usuario ya existente en Dataverse

Componentes de la solución

Si no existe, crear nueva solución para guardar los componentes

Entidad:

Nombre	Nombre lógico
Usuario / Paciente	user / mra_patient

Campos:

Nombre	Nombre lógico	Tipo de datos
Nombre	mra_name	String(100)
Apellidos	mra_surname	String(400)
DNI	mra_dni	String(100)
Profesión	mra_profession	String(200)
Dirección	mra_address	String(400)
Número de teléfono	mra_phonenumber	Whole Number
Email	mra_email	String(100)
Alergias	mra_allergies	String(800)
Enfermedades	mra_diseases	String(800)
Diagnósticos	mra_diagnosis	String(800)
Tratamientos	mra_treatments	String(800)
Comentarios	mra_comments	String(2000)

Nombre	S1.002 - Almacenar datos sobre la cita
Escenario	**Como** aplicación de gestión clínica **quiero** definir una nueva entidad **para** poder almacenar datos sobre las citas
Descripción	La aplicación debe ser capaz de contener los datos de las citas de los pacientes que asisten a la clínica, para ello hay que identificar los datos que es necesario almacenar y el tipo de datos. Posteriormente también deberá considerarse la relación entre la información de las citas y otra información necesaria para el correcto funcionamiento de la aplicación.
Criterios de Aceptación	**Dado que** se necesita almacenar datos sobre las citas **cuando** un paciente me pida una cita **entonces** voy a poder registrar la información relacionada con las citas dentro de la aplicación
Diseño técnico	Power apps dispone de un servicio de base de datos llamado Dataverse, usando Dataverse podemos almacenar todos los datos de las citas para posteriormente usarlos en nuestra aplicación. Una cita tendrá como mínimo los siguientes datos: ● Paciente que asistirá a la cita ● Fecha y hora de la cita ● Tipo de la cita (consulta, tratamiento, seguimiento) ● Duración de la cita ● Estado de la cita (próxima, pasada y cancelada)

<table>
<tr><td></td><td>

Diagnósticos
Tratamientos
Imágenes (fotos del tratamiento o diagnósticos)
Observaciones

Evaluar si es necesario crear una nueva entidad, de lo contrario, usar la entidad Cita ya existente en Dataverse
</td></tr>
<tr><td>Componentes de la solución</td><td>
Entidad:
<table>
<tr><th>Nombre</th><th>Nombre lógico</th></tr>
<tr><td>Appointment / Cita</td><td>appointment / mra_cita</td></tr>
</table>
Campos:
<table>
<tr><th>Nombre</th><th>Nombre lógico</th><th>Tipo de datos</th></tr>
<tr><td>Paciente</td><td>mra_contact</td><td>Lookup</td></tr>
<tr><td>Fecha y hora</td><td>mra_dateandtime</td><td>Date and Time</td></tr>
<tr><td>Tipo de cita</td><td>mra_type</td><td>Option Set</td></tr>
<tr><td>Duración</td><td>mra_duration</td><td>duration</td></tr>
<tr><td>Estado de la cita</td><td>status reason</td><td>Option set</td></tr>
<tr><td>Tratamiento</td><td>mra_treatments</td><td>Lookup</td></tr>
<tr><td>Diagnósticos</td><td>mra_diagnosis</td><td>Lookup</td></tr>
<tr><td>Imagenes</td><td>mra_images</td><td>Image</td></tr>
<tr><td>Comentarios</td><td>mra_comments</td><td>String(2000)</td></tr>
</table>
</td></tr>
</table>

Nombre	S1.003 - Almacenar datos sobre los tratamientos
Escenario	**Como** aplicación de gestión clínica **quiero** definir una nueva entidad **para** poder almacenar datos sobre los tratamientos
Descripción	La aplicación debe ser capaz de contener los datos de los tratamientos de los pacientes que asisten a la clínica, para ello hay que identificar los datos que es necesario almacenar y el tipo de datos. Posteriormente también deberá considerarse la relación entre la información de los tratamientos y otra información necesaria para el correcto funcionamiento de la aplicación.
Criterios de Aceptación	**Dado que** se necesita almacenar datos sobre los tratamientos **cuando** un paciente recibe un tratamiento **entonces** voy a poder registrar la información relacionada con los tratamientos dentro de la aplicación
Diseño técnico	Power apps dispone de un servicio de base de datos llamado Dataverse, usando Dataverse podemos almacenar todos los datos de los tratamientos para posteriormente usarlos en nuestra aplicación. Un tratamiento tendrá como mínimo los siguientes datos: ● Paciente al que se le ha hecho el tratamiento ● Cita donde se ha realizado el tratamiento ● Tipo de tratamiento ● Duración del tratamiento ● Descripción

<table>
<tr><td></td><td>

Requiere autorización
Documentos de autorización
Imágenes (fotos del tratamiento)
Observaciones

Crear una nueva entidad llamada Tratamientos.
</td></tr>
<tr><td>Componentes de la solución</td><td>
Entidad:
<table>
<tr><th>Nombre</th><th>Nombre lógico</th></tr>
<tr><td>Tratamientos</td><td>mra_treatments</td></tr>
</table>
Campos:
<table>
<tr><th>Nombre</th><th>Nombre lógico</th><th>Tipo de datos</th></tr>
<tr><td>Paciente</td><td>mra_contact</td><td>Lookup</td></tr>
<tr><td>Cita</td><td>appointment</td><td>Lookup</td></tr>
<tr><td>Tipo de tratamiento</td><td>mra_type</td><td>Option Set</td></tr>
<tr><td>Duración</td><td>mra_duration</td><td>Duration</td></tr>
<tr><td>Descripción</td><td>mra_description</td><td>String(2000)</td></tr>
<tr><td>Requiere autorización</td><td>mra_isauthorizationrequired</td><td>Yes/No</td></tr>
<tr><td>Documentos de autorización</td><td>mra_attatchment</td><td>Files</td></tr>
<tr><td>Imagenes</td><td>mra_images</td><td>Image</td></tr>
<tr><td>Comentarios</td><td>mra_comments</td><td>String(2000)</td></tr>
</table>
</td></tr>
</table>

Nombre	S1.004 - Almacenar datos sobre los Diagnósticos
Escenario	**Como** aplicación de gestión clínica **quiero** definir una nueva entidad **para** poder almacenar datos sobre los Diagnósticos
Descripción	La aplicación debe ser capaz de contener los datos de los diagnósticos de los pacientes que asisten a la clínica, para ello hay que identificar los datos que es necesario almacenar y el tipo de datos. Posteriormente también deberá considerarse la relación entre la información de los diagnósticos y otra información necesaria para el correcto funcionamiento de la aplicación.
Criterios de Aceptación	**Dado que** se necesita almacenar datos sobre los Diagnósticos **cuando** un paciente es diagnosticado **entonces** voy a poder registrar la información relacionada con los diagnósticos dentro de la aplicación
Diseño técnico	Power apps dispone de un servicio de base de datos llamado Dataverse, usando Dataverse podemos almacenar todos los datos de los diagnósticos para posteriormente usarlos en nuestra aplicación. Un Diagnóstico tendrá como mínimo los siguientes datos: ● Paciente diagnosticado ● Cita donde se ha realizado el Diagnósticos ● Tipo de Diagnóstico

<table>
<tr><td></td><td>

Descripción
Imágenes (fotos diagnóstico)
Observaciones

Crear una nueva entidad llamada Diagnósticos
</td></tr>
<tr><td>Componentes de la solución</td><td>
Entidad:
<table>
<tr><th>Nombre</th><th>Nombre lógico</th></tr>
<tr><td>Diagnósticos</td><td>mra_Diagnósticos</td></tr>
</table>
Campos:
<table>
<tr><th>Nombre</th><th>Nombre lógico</th><th>Tipo de datos</th></tr>
<tr><td>Paciente</td><td>mra_contact</td><td>Lookup</td></tr>
<tr><td>Cita</td><td>appointment</td><td>Lookup</td></tr>
<tr><td>Tipo de Diagnósticos</td><td>mra_type</td><td>Option Set</td></tr>
<tr><td>Descripción</td><td>mra_description</td><td>String(2000)</td></tr>
<tr><td>Imagenes</td><td>mra_images</td><td>Image</td></tr>
<tr><td>Comentarios</td><td>mra_comments</td><td>String(2000)</td></tr>
</table>
</td></tr>
</table>

4.2.2 Evaluación del desarrollo del primer sprint

Una vez concluido el primer sprint o ciclo de nuestro proyecto nos disponemos a analizar los resultados obtenidos y las dificultades encontradas durante el desarrollo, así como el proceso que se ha seguido para el mismo. Primero, se revisan los objetivos y los requisitos del proyecto, y se comparan con los resultados; si se han encontrado desviaciones significativas, es posible que se deba realizar

una reevaluación del plan general y ajustar los objetivos y los requisitos en consecuencia.

Para empezar, el objetivo de este sprint o iteración era principalmente iniciar la aplicación y aportar las estructuras de datos necesarias para almacenar la información. Todos los pasos que habrían que realizarse para inicializar el proyecto y la aplicación, que usaremos como núcleo alrededor de la cual orbitan el resto de componentes que iremos añadiendo, pueden consultarse en la sección de *"Iniciación del proyecto y aplicación"*. Con respecto al diseño del modelo de datos, este también puede encontrarse en la misma sección mencionada anteriormente; a partir de esta perspectiva en alto nivel de los datos se pudieron extraer los requisitos para el primer sprint.

En el desarrollo del primer requisito, *S1.001*, se han añadido todos los atributos definidos agregando también la capacidad de almacenar datos sobre la fecha de nacimiento del paciente, su número de teléfono fijo y los detalles de la dirección tales como: localidad, municipio, país, código postal, nombre de la calle, número del portal y piso. No fue necesario crear una nueva entidad ya que ha sido posible usar la entidad OOB, Contact, que nos ofrece Dataverse, para acomodar todos los datos del paciente. Podemos observar en la Figura 10 donde hemos hecho uso de los atributos generados por defecto que nos ofrece Dataverse y donde hemos necesitado una mayor personalización advirtiendo que aquellos que hemos creado por nuestra cuenta tienen el prefijo de nuestro editor en el nombre lógico del atributo.

Gestion Clinic APP > Tables > Contact > **Columns**

Display name	Name	Data type	Managed	Customizable	Required	Searchable
Alergias	mra_Alergias	Text area	No	Yes	No	Yes
Apellidos	LastName	Single line of text	Yes	Yes	Yes	Yes
Correo electrónico	EMailAddress1	Email	Yes	Yes	No	Yes
Código Postal	Address1_PostalCode	Single line of text	Yes	Yes	No	Yes
Dirección	Address1_Composite	Multiple lines of text	Yes	Yes	No	Yes
Dirección: nombre de la calle y número	Address1_Line1	Single line of text	Yes	Yes	No	Yes
Dirección: portal, edificio, piso	Address1_Line2	Single line of text	Yes	Yes	No	Yes
DNI	mra_DNI	Single line of text	No	Yes	No	Yes
Fecha de nacimiento	BirthDate	Date only	Yes	Yes	No	Yes
Localidad	Address1_County	Single line of text	Yes	Yes	No	Yes
Municipio	Address1_City	Single line of text	Yes	Yes	No	Yes
Método de Contacto Preferente	PreferredContactMethodC...	Choice	Yes	Yes	No	Yes
Nombre	FirstName	Single line of text	Yes	Yes	Yes	Yes
Nombre Completo Primary name column	FullName	Single line of text	Yes	Yes	No	Yes
País	Address1_Country	Single line of text	Yes	Yes	No	Yes
Profesión	JobTitle	Single line of text	Yes	Yes	No	Yes
Teléfono fijo	Address1_Telephone1	Phone number	Yes	Yes	No	Yes
Teléfono móvil	MobilePhone	Phone number	Yes	Yes	No	Yes

Figura 10. Atributos de la entidad Paciente

Con respecto al requisito *S1.002*, igual que con el requisito anterior, podemos ver en la Figura 11 la combinación entre elementos ya provistos por las entidades OOB de Dataverse y nuestras propias personalizaciones usando nuestro prefijo de editor.

Gestion Clinic APP > Tables > Appointment > **Columns**

Display name	Name	Data type	Managed	Customizable	Required	Searchable
Descripción	Description	Multiple lines of text	Yes	Yes	No	Yes
Duración	ScheduledDurationMinutes	Duration	Yes	Yes	No	Yes
Estado	StateCode	Choice	Yes	Yes	Yes	Yes
Hora Fin	ScheduledEnd	Date and time	Yes	Yes	Yes	Yes
Hora Inicio	ScheduledStart	Date and time	Yes	Yes	Yes	Yes
Paciente	RegardingObjectId	Lookup	Yes	Yes	No	Yes
Status Reason	StatusCode	Choice	Yes	Yes	No	Yes
Tipo de cita	mra_Tipodecita	Choice	No	Yes	No	Yes

Figura 11. Atributos de la entidad Cita

Tanto para los requisitos *S1.003* y *S1.004* hemos necesitado crear nuevas entidades personalizadas. Creándolas desde cero Dataverse genera una serie de campos

básicos como la id, el nombre, la fecha de creación y el creador de cada dato, última fecha de modificación y persona que hizo la modificación o el estado, como se muestra en las Figuras 12 y 13.

Gestion Clinic APP > Tables > Treatment > **Columns**

Display name	Name	Data type	Managed	Customizable	Required	Searchable
Created By	CreatedBy	Lookup	No	Yes	No	Yes
Created On	CreatedOn	Date and time	No	Yes	No	Yes
Created By (Delegate)	CreatedOnBehalfBy	Lookup	No	Yes	No	Yes
Import Sequence Number	ImportSequenceNumber	Whole number	No	Yes	No	Yes
Modified By	ModifiedBy	Lookup	No	Yes	No	Yes
Modified On	ModifiedOn	Date and time	No	Yes	No	Yes
Modified By (Delegate)	ModifiedOnBehalfBy	Lookup	No	Yes	No	Yes
Cita	mra_appointment	Lookup	No	Yes	No	Yes
Documento de autorización	mra_autorizacion	File	No	Yes	No	No
Paciente	mra_contact	Lookup	No	Yes	No	Yes
Descripción	mra_Descripcion	Text area	No	Yes	No	Yes
Duración	mra_Duracion	Duration	No	Yes	No	Yes
Requiere autorización	mra_isauthorizationrequired	Yes/no	No	Yes	No	Yes
Name Primary name column	mra_Name	Single line of text	No	Yes	Yes	Yes
Tipo de tratamiento	mra_Tipodetratamiento	Choice	No	Yes	No	Yes
Treatment	mra_treatmentId	Unique identifier	No	Yes	Yes	Yes

Figura 12. Atributos de la entidad Tratamiento

Gestion Clinic APP > Tables > Diagnosis > **Columns**

Display name	Name	Data type	Managed	Customizable	Required	Searchable
Created By	CreatedBy	Lookup	No	Yes	No	Yes
Created On	CreatedOn	Date and time	No	Yes	No	Yes
Created By (Delegate)	CreatedOnBehalfBy	Lookup	No	Yes	No	Yes
Import Sequence Number	ImportSequenceNumber	Whole number	No	Yes	No	Yes
Modified By	ModifiedBy	Lookup	No	Yes	No	Yes
Modified On	ModifiedOn	Date and time	No	Yes	No	Yes
Modified By (Delegate)	ModifiedOnBehalfBy	Lookup	No	Yes	No	Yes
Cita	mra_Appointment	Lookup	No	Yes	No	Yes
Paciente	mra_contact	Lookup	No	Yes	No	Yes
Descripción	mra_Descripcion	Text area	No	Yes	No	Yes
Diagnosis	mra_diagnosisId	Unique identifier	No	Yes	Yes	Yes
Name Primary name column	mra_name	Single line of text	No	Yes	Yes	Yes
Tipo de diagnosis	mra_Tipodediagnosis	Choice	No	Yes	No	Yes
Organization Id	OrganizationId	Lookup	No	Yes	No	No
Record Created On	OverriddenCreatedOn	Date only	No	Yes	No	Yes
Status	statecode	Choice	No	Yes	Yes	Yes

Figura 13. Atributos de la entidad Diagnósticos

4.3 Desarrollo del segundo sprint

Una vez tenemos el modelo de datos implementado podemos centrarnos en el aspecto gráfico de nuestra aplicación. De hecho vamos a poner especial hincapié en elementos como los formularios, teniendo en cuenta los diferentes campos que tenemos, su colocación, la facilidad de visualización de estos para el usuario, qué datos será obligatorios rellenar y cuáles opcionales; y las vistas de esos datos, que aportarán una visión concreta según parámetros predefinidos.

4.3.1 Requisitos para el segundo sprint

Nombre	S2.001 - Formulario principal de la entidad Cita
Escenario	**Como** usuario de la aplicación de la clínica **quiero** disponer de un formulario con los datos de citas **para** crear, editar o ver una cita
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Citas, podrá seleccionar una cita en concreto para ver o editar los datos o usar el botón de crear nueva cita, entonces se abrirá el formulario principal de la entidad Cita, el cual tendrá todos los datos necesarios, y además estos estarán colocados de tal manera que sea legible e intuitivo para el usuario.
Criterios de Aceptación	**Dado que** el usuario quiere ver o editar datos de citas **cuando** seleccione una cita concreta desde la vista **entonces** se abrirá el formulario principal de Citas **y** **Dado que** el usuario quiere crear una nueva cita **cuando** seleccione el botón “Cita” **entonces** se abrirá el formulario principal de Citas

<table>
<tr><td>Diseño técnico</td><td>Dado que la entidad Citas de nuestro sistema viene de la entidad OOB Appointment, también disponemos de un primer acercamiento al formulario de cita provisto por Dataverse. El objetivo en este requisito será editar el formulario principal ya existente para la entidad Appointment, eliminando los datos que existen para esa entidad que no vayamos a usar y añadiendo los datos nuevos que hemos construido sobre ella gracias a la personalización. Cambiando la distribución de los datos para mejorar la experiencia de usuario y añadiendo que campos son opcionales y cuáles obligatorios dentro del formulario.</td></tr>
<tr><td>Componentes de la solución</td><td>Formulario principal de la entidad Appointment, cambiar el nombre del formulario para que tenga el mismo nombre que la entidad.
El formulario tendrá los siguientes datos:
<table><tr><th>Nombre</th><th>Nombre Lógico</th><th>Requerido</th></tr><tr><td>Nombre</td><td>subject</td><td>No</td></tr><tr><td>estado</td><td>statecode</td><td>Sí</td></tr><tr><td>Paciente</td><td>regardingobjectid</td><td>Sí</td></tr><tr><td>Tipo de cita</td><td>mra_tipodecita</td><td>No</td></tr><tr><td>Hora Inicio (incluye fecha)</td><td>scheduledstart</td><td>Sí</td></tr><tr><td>Hora Fin (incluye fecha</td><td>scheduledend</td><td>Sí</td></tr></table></td></tr>
</table>

Duración	scheduledduration minutes	No

Algunos de los datos no serán requeridos para el usuario ya que estos se rellenarán de forma automatizada, bien sea porque esté implementado como parte de los componentes que ya ofrece el sistema o porque lo implementemos más adelante.

Y además incluirá las siguientes secciones:

- Timeline, donde ver el historial de acciones ocurridas con respecto a la cita y poder agregar notas
- Tratamientos, donde se verá una vista básica de los tratamientos relacionados con la cita, desde ahí se podrá acceder a los datos concretos de un tratamiento o crear uno nuevo para la propia cita.
- Diagnóstico, donde se verá una vista básica de los diagnósticos relacionados con la cita, desde ahí se podrá acceder a los datos concretos de un diagnóstico o crear uno nuevo para la propia cita.
- Adjuntos, sección donde es posible ver y agregar archivos adjuntos a la cita
- Descripción, un campo de texto editable para añadir de forma complementaria a la información de la cita

Las vistas necesarias para visualizar los datos de tratamientos y diagnósticos se crearán como parte de otro requisito.

Nombre	S2.002 - Formulario principal de la entidad Paciente
Escenario	**Como** usuario de la aplicación de la clínica **quiero** disponer de un formulario con los datos de un paciente **para** crear, editar o ver un paciente
Descripción	El usuario al entrar a la aplicación podrá seleccionar el área de datos de Pacientes en el menú lateral, accediendo a las vistas de pacientes, seleccionando un paciente en concreto podrá ver o editar los datos o usar el botón de crear nuevo paciente, entonces se abrirá el formulario principal de la entidad Paciente. Este tendrá todos los datos necesarios y además ubicados de tal manera que sea legible e intuitivo para el usuario.
Criterios de Aceptación	**Dado que** el usuario quiere ver o editar datos de pacientes **cuando** seleccione un paciente concreto desde la vista **entonces** se abrirá el formulario principal de Pacientes **y** **Dado que** el usuario quiere crear un nuevo Paciente **cuando** seleccione el botón “Nuevo” **entonces** se abrirá el formulario principal de Pacientes
Diseño técnico	Dado que la entidad Pacientes de nuestro sistema viene de la entidad OOB Contact, también disponemos de un primer acercamiento al formulario de pacientes provisto por Dataverse. El objetivo en este requisito será editar el formulario principal ya existente para la entidad Contact, eliminando los datos ya existentes que no vayamos a usar y añadiendo los datos nuevos que hemos

<table>
<tr><td></td><td>construido sobre ella gracias a la personalización. Cambiando además la distribución de los datos para mejorar la experiencia de usuario y añadiendo que campos son opcionales y cuáles obligatorios dentro del formulario.</td></tr>
<tr><td>Componentes de la solución</td><td>Formulario principal de la entidad Contact, cambiar el nombre del formulario para que tenga el mismo nombre que la entidad, en nuestro caso Pacientes.

Dentro del formulario tendremos varias pestañas para los distintos conjuntos de datos, los datos principales se colocarán en la primera pestaña, “Datos del Paciente”; los datos de tratamientos y diagnósticos se ubicarán en la siguiente pestaña, Datos Clínicos; y los datos de las citas del paciente en la última pestaña, “Citas”.

La primera pestaña tendrá los siguientes datos:
<table><tr><th>Nombre</th><th>Nombre Lógico</th><th>Requerido</th></tr><tr><td>Nombre</td><td>firstname</td><td>Sí</td></tr><tr><td>Apellidos</td><td>lastname</td><td>Sí</td></tr><tr><td>DNI</td><td>mra_DNI</td><td>Sí</td></tr><tr><td>Fecha de nacimiento</td><td>birthdate</td><td>Sí</td></tr><tr><td>Profesión</td><td>jobtitle</td><td>No</td></tr></table></td></tr>
</table>

Dirección: nombre de la calle y número	address1_line1	No
Dirección: portal, edificio, piso	address1_line2	No
Código Postal	address1_postalcode	No
Localidad	address1_county	No
Provincia	address1_city	No
País	address1_country	No
Método de Contacto Preferente	preferredcontactmethodcode	No
Teléfono móvil	mobilephone	No
Teléfono fijo	address1_telephone1	No
Correo electrónico	emailaddress1	No
Enfermedades	mra_enfermedades	No
Alergias	mra_Alergias	No

Y además incluirá las siguientes secciones:

- Observaciones, donde ver el historial de acciones ocurridas con respecto al paciente y poder agregar notas

	• Datos personales, donde encontraremos campos como: nombre, apellidos, DNI, profesión, fecha de nacimiento, enfermedades o alergias y datos de dirección • Datos de Contacto, con los campos: método de contacto preferente, teléfono móvil, teléfono fijo y correo electrónico La pestaña de Datos Clínicos tendrá la información de los tratamientos y diagnósticos de los pacientes; estos datos se presentarán de la misma manera que la vista principal de cada entidad. Finalmente, la pestaña de Citas nos permitirá ver todas las citas que ha tenido el paciente ya sea que se hayan vencido las citas o no, para ello usaremos la vista correspondiente. Todas las vistas mencionadas se desarrollarán como parte de otros requisitos en esta iteración.

Nombre	S2.003 - Formulario principal de la entidad Diagnóstico
Escenario	**Como** usuario de la aplicación de la clínica **quiero** disponer de un formulario con los datos de diagnósticos **para** crear, editar o ver un diagnóstico concreto
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Diagnósticos, podrá seleccionar uno en concreto para ver o editar los datos o usar el

<table>
<tr><td></td><td>botón de nuevo, entonces se abrirá el formulario principal de la entidad, el cual tendrá todos los datos necesarios, y además estos estarán colocados de tal manera que sea legible e intuitivo para el usuario.</td></tr>
<tr><td>Criterios de Aceptación</td><td>Dado que el usuario quiere ver o editar datos de diagnósticos
cuando seleccione un diagnóstico concreto desde la vista
entonces se abrirá el formulario principal de Diagnóstico
y
Dado que el usuario quiere crear un nuevo diagnóstico
cuando seleccione el botón “Nuevo”
entonces se abrirá el formulario principal de diagnóstico</td></tr>
<tr><td>Diseño técnico</td><td>Al ser una entidad personalizada, todos los campos han sido creados de cero, usaremos el componente formulario creado por defecto en Dataverse para incluir todos los campos.</td></tr>
<tr><td>Componentes de la solución</td><td>Formulario principal de la entidad Diagnóstico, cambiar el nombre del formulario para que tenga el mismo nombre que la entidad.
El formulario tendrá los siguientes datos:
<table>
<tr><th>Nombre</th><th>Nombre Lógico</th><th>Requerido</th></tr>
<tr><td>Nombre</td><td>mra_name</td><td>No</td></tr>
<tr><td>Tipo de diagnóstico</td><td>mra_tipodediagnosis</td><td>No</td></tr>
<tr><td>Paciente</td><td>mra_contact</td><td>Sí</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td>Cita</td><td>mra_appointment</td><td>Sí</td></tr><tr><td>Descripción</td><td>mra_descripcion</td><td>No</td></tr></table> Algunos de los datos no serán requeridos para el usuario ya que estos se rellenarán de forma automatizada, bien sea que esté implementado como parte de los componentes porque ya ofrece el sistema o porque lo implementemos más adelante. Y además incluirá las siguientes secciones: • Timeline, donde ver el historial de acciones ocurridas con respecto al diagnóstico y poder agregar notas.

Nombre	S2.004 - Formulario principal de la entidad Tratamientos
Escenario	**Como** usuario de la aplicación de la clínica **quiero** disponer de un formulario con los datos de tratamientos **para** crear, editar o ver un tratamiento concreto
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Tratamientos, podrá seleccionar uno en concreto para ver o editar los datos o usar el botón de nuevo, entonces se abrirá el formulario principal de la entidad, el cual tendrá todos los datos necesarios, y además estos estarán colocados de tal manera que sea legible e intuitivo para el usuario.

<table>
<tr><td>Criterios de Aceptación</td><td>Dado que el usuario quiere ver o editar datos de tratamientos
cuando seleccione un tratamiento concreto desde la vista
entonces se abrirá el formulario principal de Tratamiento
y
Dado que el usuario quiere crear un nuevo tratamiento
cuando seleccione el botón “Nuevo”
entonces se abrirá el formulario principal de tratamiento</td></tr>
<tr><td>Diseño técnico</td><td>Al ser una entidad personalizada, todos los campos han sido creados de cero, usaremos el componente formulario creado por defecto en Dataverse para incluir todos los campos.</td></tr>
<tr><td>Componentes de la solución</td><td>Formulario principal de la entidad Tratamiento, cambiar el nombre del formulario para que tenga el mismo nombre que la entidad.
El formulario tendrá los siguientes datos:
<table><tr><th>Nombre</th><th>Nombre Lógico</th><th>Requerido</th></tr><tr><td>Nombre</td><td>mra_name</td><td>No</td></tr><tr><td>Tipo de Tratamiento</td><td>mra_tipodetratamiento</td><td>No</td></tr><tr><td>Paciente</td><td>mra_contact</td><td>Sí</td></tr><tr><td>Cita</td><td>mra_appointment</td><td>Sí</td></tr><tr><td>Descripción</td><td>mra_descripcion</td><td>No</td></tr></table></td></tr>
</table>

	<table><tr><td>Requiere autorización</td><td>mra_isauthorization required</td><td>Sí</td></tr><tr><td>Documento de autorización</td><td>mra_autorizacion</td><td>No</td></tr></table> Algunos de los datos no serán requeridos para el usuario ya que estos se rellenarán de forma automatizada, bien que esté implementado como parte de los componentes porque ya ofrece el sistema o porquelo implementemos más adelante. Y además incluirá las siguientes secciones: ● Timeline, donde ver el historial de acciones ocurridas con respecto al tratamiento y poder agregar notas.

Nombre	S2.005 - Vistas de la entidad Citas
Escenario	**Como** usuario de la aplicación de la clínica **quiero** ver los datos de citas en vistas **para** disponer de ellos de forma ordenada y poder filtrar conjuntos de datos concretos
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Citas, podrá seleccionar entre varias vistas distintas (con filtros aplicados sobre los datos), que mostrarán una lista de citas, incluyendo los atributos más esenciales, para la adecuada visualización e identificación de los datos.

Criterios de Aceptación	**Dado que** el usuario quiere ver datos de citas **cuando** seleccione el área de Citas en el menú lateral **entonces** dispondrá de varias vistas que mostrarán los datos de citas según diferentes filtros **y** **Dado que** el usuario quiere identificar los datos de citas **cuando** seleccione el área de Citas en el menú lateral y esté viendo una de las vistas de la entidad **entonces** será capaz de distinguir cada dato y ver un resumen de la información más importante
Diseño técnico	Al ser una entidad OOB, hay muchas vistas que Dataverse nos ofrece por defecto, seleccionaremos las que se aplican a nuestro caso y eliminaremos las demás de nuestra solución. Si fuera necesario, si hay alguna vista que queramos tener que Dataverse no nos ofrezca, se crearía una vista personalizada para la entidad Appointment. También editaremos las vistas ya existentes para que usen los atributos más relevantes dentro de los datos que hemos seleccionado.
Componentes de la solución	Lo primero sería definir el conjunto de atributos que queremos mostrar para esta entidad en concreto y asegurar que todas las vistas de la entidad funcionan usando esos atributos para así tener una mejor experiencia de usuario.

Estos atributos serán los siguientes y en ese orden:

Nombre	Nombre Lógico	Tamaño de columna px
Nombre	subject	300px
Paciente	regardingobjectid	150px
Hora Inicio (incluye fecha)	scheduledstart	100px
Hora Fin (incluye fecha	scheduledend	100px
Duración	scheduledduration minutes	75px
Tipo de cita	mra_tipodecita	75px
Estado	statecode	100px

Las vistas que tendremos serán las siguientes:

- Todas las citas, sin filtro se muestran todas las citas con datos. Vista por defecto.
- Citas pasadas, muestra las citas completadas, no canceladas.
- Citas canceladas, muestra las citas que han sido canceladas.
- Mis citas próximas, muestra las citas que aún no han vencido.
- Citas últimos 30 días, muestra las citas de los últimos 30 días.

Nombre	S2.006 - Vistas de la entidad Pacientes
Escenario	**Como** usuario de la aplicación de la clínica **quiero** ver los datos de pacientes en vistas **para** disponer de ellos de forma ordenada y poder filtrar conjuntos de datos concretos
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Pacientes, podrá ver la vista por defecto, que mostrarán una lista de pacientes, incluyendo los atributos más esenciales, para la adecuada visualización e identificación de los datos.
Criterios de Aceptación	**Dado que** el usuario quiere ver datos de pacientes **cuando** seleccione el área de Pacientes en el menú lateral **entonces** verá la vista por defecto de pacientes **y** **Dado que** el usuario quiere identificar los datos de pacientes **cuando** seleccione el área de Pacientes en el menú lateral y esté viendo una de las vistas de la entidad **entonces** será capaz de distinguir cada dato y ver un resumen de la información más importante
Diseño técnico	Al ser una entidad OOB, hay muchas vistas que Dataverse nos ofrece por defecto, seleccionaremos la que se aplican a nuestro caso y eliminaremos las demás de nuestra solución. Si fuera necesario, si hay alguna vista que queramos tener que Dataverse no nos ofrezca, se crearía una vista personalizada para la entidad Contact.

<table>
<tr><td></td><td>También en el caso de usar las vistas ya existentes, las editaremos para que usen los atributos más relevantes dentro de los datos que hemos seleccionado.</td></tr>
<tr><td>Componentes de la solución</td><td>Lo primero sería definir el conjunto de atributos que queremos mostrar para esta entidad en concreto y asegurar que todas las vistas de la entidad funcionan usando esos atributos para así tener una mejor experiencia de usuario.

Estos atributos serán los siguientes y en ese orden:
<table>
<tr><th>Nombre</th><th>Nombre Lógico</th><th>Tamaño de columna px</th></tr>
<tr><td>Nombre completo</td><td>fullname</td><td>250px</td></tr>
<tr><td>Profesión</td><td>jobtitle</td><td>100px</td></tr>
<tr><td>DNI</td><td>mra_dni</td><td>75px</td></tr>
<tr><td>Método de Contacto Preferente</td><td>preferredcontactmethodcode</td><td>125px</td></tr>
<tr><td>Teléfono móvil</td><td>mobilephone</td><td>100px</td></tr>
<tr><td>Teléfono fijo</td><td>address1_telephone1</td><td>100px</td></tr>
<tr><td>Correo electrónico</td><td>emailaddress1</td><td>150px</td></tr>
</table>
Para esta entidad solo tendremos una vista que se llamará Pacientes.</td></tr>
</table>

Nombre	S2.007 - Vistas de la entidad Diagnósticos
Escenario	**Como** usuario de la aplicación de la clínica **quiero** ver los datos de diagnósticos en vistas **para** disponer de ellos de forma ordenada y poder filtrar conjuntos de datos concretos
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Diagnósticos, podrá ver la vista por defecto, que mostrarán una lista de diagnósticos, incluyendo los atributos más esenciales, para la adecuada visualización e identificación de los datos.
Criterios de Aceptación	**Dado que** el usuario quiere ver datos de diagnósticos **cuando** seleccione el área de Diagnósticos en el menú lateral **entonces** verá la vista por defecto de diagnósticos **y** **Dado que** el usuario quiere identificar los datos de diagnósticos **cuando** seleccione el área de Diagnósticos en el menú lateral y esté viendo una de las vistas de la entidad **entonces** será capaz de distinguir cada dato y ver un resumen de la información más importante
Diseño técnico	Siendo una entidad personalizada depende de nosotros la creación de nuevas vistas, aun así, Dataverse ya ha creado la vista por defecto por nosotros, la editaremos para que use los atributos más relevantes dentro de los datos que hemos seleccionado.

<table>
<tr><td>Componentes de la solución</td><td>Lo primero sería definir el conjunto de atributos que queremos mostrar para esta entidad en concreto.
Estos atributos serán los siguientes y en ese orden:
<table><tr><th>Nombre</th><th>Nombre Lógico</th><th>Tamaño de columna px</th></tr><tr><td>Nombre</td><td>mra_name</td><td>300px</td></tr><tr><td>Paciente</td><td>mra_contact</td><td>150px</td></tr><tr><td>Cita</td><td>mra_appointment</td><td>300px</td></tr><tr><td>Tipo de diagnóstico</td><td>mra_tipodediagnosis</td><td>150px</td></tr><tr><td>Creado en</td><td>createdon</td><td>100px</td></tr></table>
Para esta entidad solo tendremos una vista que se llamará Diagnósticos.</td></tr>
</table>

Nombre	S2.008 - Vistas de la entidad tratamientos
Escenario	**Como** usuario de la aplicación de la clínica **quiero** ver los datos de tratamientos en vistas **para** disponer de ellos de forma ordenada y poder filtrar conjuntos de datos concretos
Descripción	El usuario al entrar a la aplicación y navegar desde el menú lateral al área de Tratamientos, podrá ver la vista por defecto, que mostrarán una lista de tratamientos, incluyendo los atributos más esenciales, para la adecuada visualización e identificación de los datos.

<table>
<tr><td>Criterios de Aceptación</td><td>Dado que el usuario quiere ver datos de tratamientos
cuando seleccione el área de Tratamientos en el menú lateral
entonces verá la vista por defecto de tratamientos
y
Dado que el usuario quiere identificar los datos de tratamientos
cuando seleccione el área de Tratamientos en el menú lateral y esté viendo una de las vistas de la entidad
entonces será capaz de distinguir cada dato y ver un resumen de la información más importante</td></tr>
<tr><td>Diseño técnico</td><td>Siendo una entidad personalizada depende de nosotros la creación de nuevas vistas, aun así, Dataverse ya ha creado la vista por defecto por nosotros, la editaremos para que use los atributos más relevantes dentro de los datos que hemos seleccionado.</td></tr>
<tr><td>Componentes de la solución</td><td>Lo primero sería definir el conjunto de atributos que queremos mostrar para esta entidad en concreto.
Estos atributos serán los siguientes y en ese orden:
<table><tr><th>Nombre</th><th>Nombre Lógico</th><th>Tamaño de columna px</th></tr><tr><td>Nombre</td><td>mra_name</td><td>300px</td></tr><tr><td>Paciente</td><td>mra_contact</td><td>150px</td></tr><tr><td>Cita</td><td>mra_appointment</td><td>300px</td></tr><tr><td>Tipo de tratamiento</td><td>mra_tipodetratamiento</td><td>150px</td></tr></table></td></tr>
</table>

	<table><tr><td>Creado en</td><td>createdon</td><td>100px</td></tr></table> Para esta entidad solo tendremos una vista que se llamará Tratamientos.

Nombre	S2.009 - Usar buscador en la ventana de vistas
Escenario	**Como** usuario de la aplicación de la clínica **quiero** usar el campo de búsqueda **para** filtrar los datos de una vista
Descripción	El usuario, al entrar a la aplicación y navegar desde el menú lateral al área de su elección, podrá ver la vista por defecto de la entidad que haya seleccionado, además de seleccionar otras vistas en el caso de que la entidad disponga de más de una. Adicionalmente a este comportamiento, también se dispondrá de una barra de búsqueda que el usuario podrá utilizar para filtrar aún más los datos.
Criterios de Aceptación	**Dado que** el usuario quiere filtrar los datos de las vistas **cuando** escriba algo en el campo de texto **entonces** se filtrarán los datos de la vista teniendo en cuenta la información de todos los campos
Diseño técnico	El objetivo de este requisito es habilitar la búsqueda para todos los campos que formen parte de las vistas, ya que por defecto el campo de búsqueda solo aplica para el atributo principal de la entidad. Para ello editaremos un componente generado por defecto que se llama “quick find view”, donde añadiremos todos los campos

	necesarios para cada entidad.
Componentes de la solución	Para cada entidad se editará su “quick find view” correspondiente habilitando los atributos de cada vista, y estas son: ● Quick Find All Appointments, para la entidad Cita, con los campos: nombre, hora fin, hora inicio, paciente, duración, estado y tipo de cita. ● Quick Find Contacts, para la entidad Paciente, con los campos: nombre completo, profesión, DNI, método de contacto preferente, teléfono móvil, teléfono fijo y correo electrónico. ● Quick Find Diagnosis, para la entidad Diagnóstico, con los campos: nombre, paciente, cita, tipo de diagnóstico y creado en. ● Quick Find Treatments, para la entidad Tratamientos, con los campos: nombre, paciente, cita, tipo de tratamiento y creado en.

4.3.2 Evaluación del desarrollo del segundo sprint

En esta iteración hemos puesto el punto de mira en el apartado visual de nuestra aplicación, centrándonos sobre todo en proveer de las interfaces necesarias para que el usuario final pueda hacer un buen manejo de los datos, a la misma vez que aseguramos su usabilidad con buenas prácticas de diseño, con el objetivo de facilitar la navegación entre todos los formularios y vistas del sistema.

Para hablar sobre la edición de los formularios usaremos como ejemplo el formulario de la entidad paciente, *S2.002* como observamos en la Figura 14. Este dispone de varias pestañas distintas a las que podemos navegar y en su pestaña

principal podemos ver la organización de los datos usando muchos de los elementos esenciales que nos provee el editor de formularios. En este caso, la pestaña principal tiene dos columnas donde están incluidos los distintos elementos del formulario, estos a su vez están divididos en secciones que nos permiten identificar rápidamente y de un vistazo la naturaleza de cada conjunto de datos.

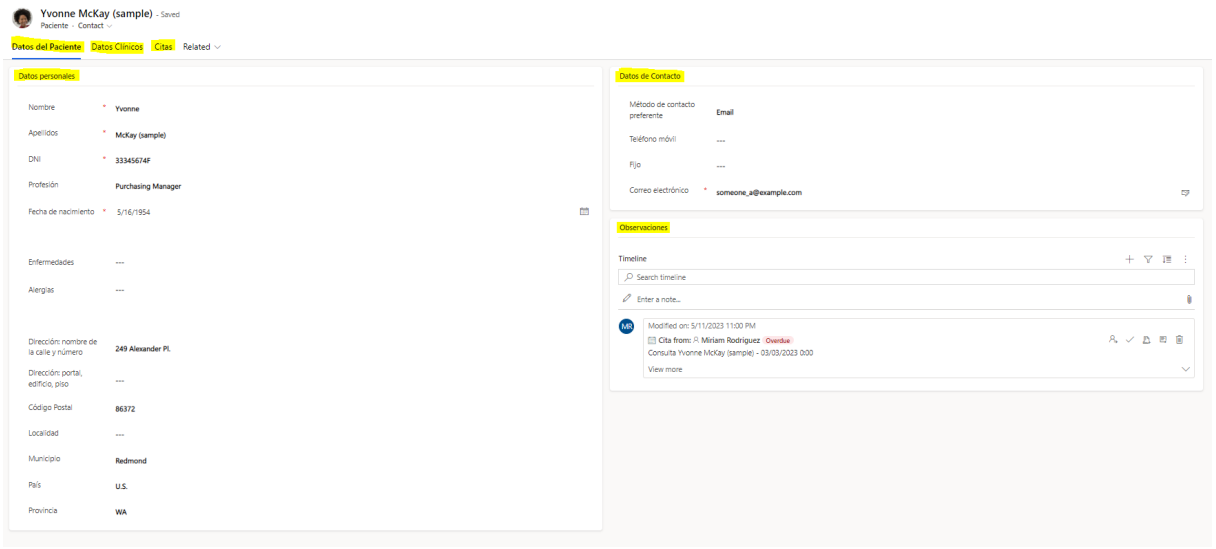

Figura 14. Formulario de la entidad Paciente.

En las otras pestañas de este formulario disponemos de las vistas para las entidades relacionadas con esta, como se puede observar en la Figura 15.

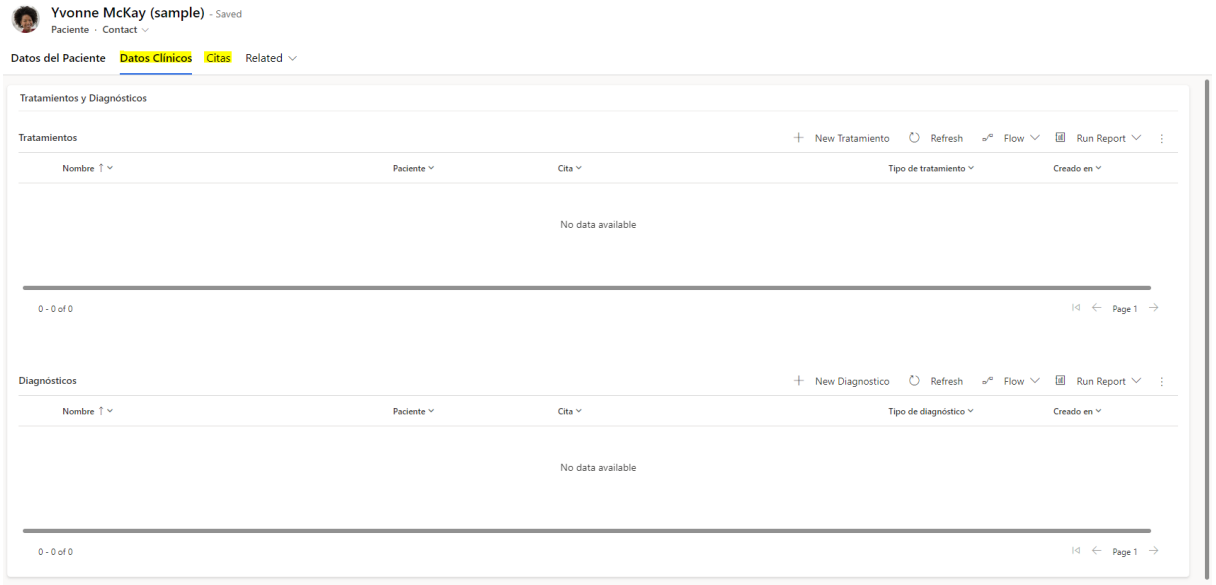

Figura 15. Pestaña Datos Clínicos del formulario de Paciente

Con respecto a las vistas, podemos ver en la Figura 16 la extensión de su uso en la entidad Citas, *S2.005*, donde disponemos de varias vistas distintas con filtros predefinidos a servicio del usuario. Estas han sido desarrolladas a beneficio de la experiencia de usuario, moderando así el uso de los filtros personalizados que también nos ofrecen las model-driven apps como parte del sistema, reduciendo el número de clics. Antes de seleccionar una vista en concreto, habrá una por defecto que se mostrará al usuario. Además, esta funcionalidad de vista por defecto es editable por el propio usuario de la aplicación sin necesidad de modificar nada de la solución o necesitar la ayuda de un desarrollador.

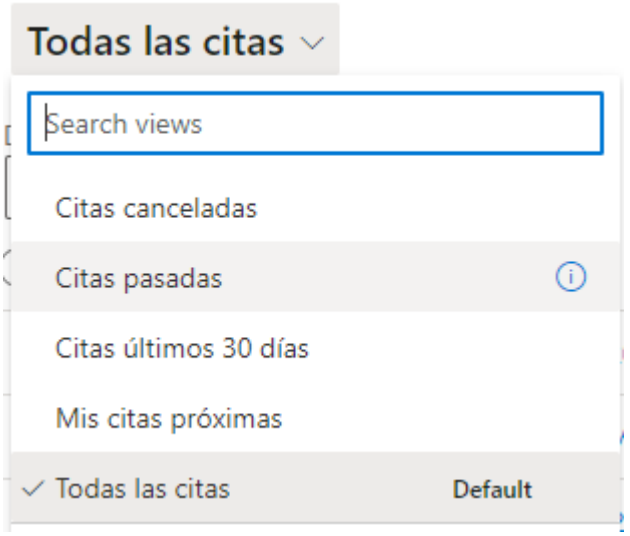

Figura 16. Selección de las distintas vistas de citas

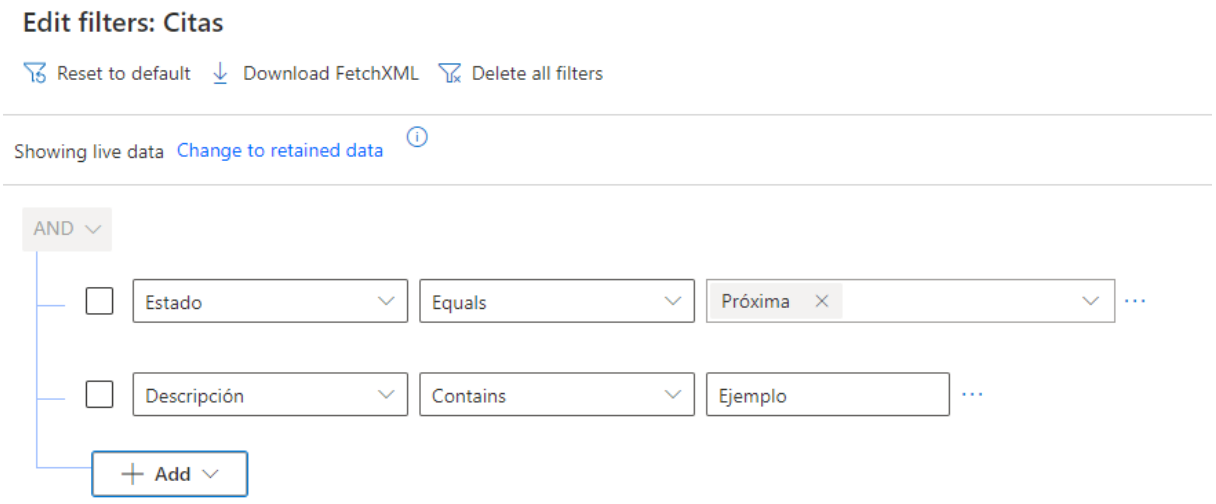

Figura 17. Pantalla de creación de filtros personalizados

Como muestra, en la Figura 17, se ha incluido la posibilidad de hacer un filtrado según los campos que aparecen en la vista usando una caja de texto, *S2.009*. Esta y todas las opciones del usuario para configurar y añadir filtros adicionales a las vistas se encuentran a la derecha de la pantalla. En la Figura 18 podemos contemplar los ejemplos creados para la vista de tratamientos.

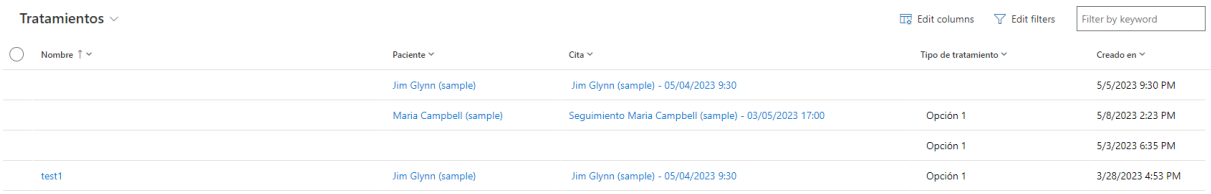

Figura 18. Vista de tratamientos

De cara a la siguiente iteración se mantiene el objetivo de mejorar la experiencia de usuario, esta vez habilitando procesos que añadan lógica a los formularios y otros que de forma automatizada por el sistema hagan un autocompletado de algunos de los datos, librando así al usuario de tener que realizar acciones repetitivas.

4.4 Desarrollo del tercer sprint

En esta iteración vamos a definir una serie de procesos automatizados, reglas de negocio y/o scripts, que van a ayudar a mejorar la lógica de nuestra aplicación. Estos procesos irán desde autocompletación de campos, revisión periódica de los datos y modificación automática, a limitaciones al rellenar los campos del formulario, su visibilidad o su obligatoriedad. Para ello usaremos flujos en la nube de Power Automate, reglas de negocio usando Power Apps y de ser necesario, scripts usando lenguaje javascript junto con la API que dispone Power Apps para su uso en formularios de model-driven apps. [24]

4.4.1 Requisitos para el tercer sprint

Nombre	S3.001 - Nueva entidad Tipos de Procedimientos
Escenario	**Como** aplicación de gestión clínica **quiero** disponer de una lista configurable de tipos de procedimientos **para** permitir que el usuario pueda reutilizar la información
Descripción	Dentro de los tratamientos o diagnósticos que se realicen en la clínica habrá algunos que ocurran de manera recurrente. Con el objetivo de ahorrar al usuario el esfuerzo de volver a introducir la misma información una y otra vez se creará una nueva entidad llamada Tipo de Procedimiento donde se podrá registrar todos estos procedimientos habituales.
Criterios de Aceptación	**Dado que** algunos procedimientos podrían ser recurrentes **cuando** se desee añadir un nuevo tipo de tratamiento o diagnóstico **entonces** el usuario podrá crear un nuevo dato en la entidad Tipo de Procedimiento **y** **Dado que** algunos procedimientos podrían ser recurrentes **cuando** se desee seleccionar un tipo de tratamiento o diagnóstico ya existente **entonces** el usuario podrá elegir entre la lista de datos de la entidad Tipo de Procedimiento

<table>
<tr><td>Diseño técnico</td><td>Esta será una nueva entidad personalizada dentro de nuestro sistema, lo que implica que habrá que crear o modificar sus componentes básicos desde cero. Primero definiremos los campos de la entidad y la añadiremos al menú principal de la aplicación, crearemos el formulario principal con estos campos, después definiremos las primeras vistas para la correcta visualización de los datos y finalmente añadiremos las relaciones de esta entidad con las entidades Tratamientos y Diagnósticos.

Una vez terminado todo lo anterior haremos los ajustes necesarios a los formularios y vistas de Tratamientos y Diagnósticos para incluir esta nueva entidad, sustituyendo el campo tipo de tratamiento o diagnóstico, respectivamente.</td></tr>
<tr><td>Componentes de la solución</td><td>Los atributos de la entidad Tipo de Procedimientos serán los siguientes:
<table><tr><th>Nombre</th><th>Nombre lógico</th><th>Tipo de datos</th></tr><tr><td>Nombre</td><td>mra_nombre</td><td>String(200)</td></tr><tr><td>Tipo</td><td>mra_tipo</td><td>Yes/No
Tratamiento/Diagnóstico</td></tr><tr><td>Descripción</td><td>mra_description</td><td>String(2000)</td></tr></table>El formulario principal tendrá el mismo nombre que la entidad y se considerará si fuera necesario añadir un formulario de creación rápida para evitar problemas de navegación en el caso de que el usuario quiera crear un</td></tr>
</table>

<table>
<tr><td></td><td>nuevo tipo de procedimiento en el momento, dentro del formulario de tratamientos o diagnósticos.
Con respecto a las vistas, esta entidad dispondrá de cuatro vistas:
Todos los tipos de procedimientos (por defecto)
Tipos de diagnósticos (tipo = diagnóstico)
Tipo de tratamientos (tipo = tratamiento)
Quick find view (para el uso del buscador)
Y estas tendrán los siguientes campos:<table><tr><th>Nombre</th><th>Nombre Lógico</th><th>Tamaño de columna px</th></tr><tr><td>Nombre</td><td>mra_nombre</td><td>150px</td></tr><tr><td>Tipo</td><td>mra_tipo</td><td>100px</td></tr><tr><td>Descripción</td><td>mra_description</td><td>300px</td></tr></table></td></tr>
</table>

Nombre	S3.002 - Autocompletar campo nombre para las citas
Escenario	**Como** aplicación de gestión clínica **quiero** que el campo nombre de la cita se rellene automáticamente **para** evitar que el usuario tenga que hacerlo manualmente
Descripción	El campo nombre servirá para identificar las distintas citas en la página de vista principal, con el objetivo de liberar al usuario de la carga de rellenar el nombre manualmente cada vez, dispondremos de un proceso

<table>
<tr><td></td><td>automatizado que haga ese trabajo por el usuario, utilizando una combinación de otros campos (obligatorios) para generar el nombre.</td></tr>
<tr><td>Criterios de Aceptación</td><td>Dado que el campo nombre está vacío por defecto
cuando se cree una nueva cita
entonces se ejecutará un proceso que rellene este campo automáticamente
y
Dado que algunos campos obligatorios de los que se extraerá el nombre son editables
cuando se edite uno de esos campos de la cita
entonces se ejecutará un proceso que actualice el nombre automáticamente</td></tr>
<tr><td>Diseño técnico</td><td>Para cumplir el objetivo de este requisito vamos a crear dos flujos de nube, uno que se lanzará con la creación de nuevas citas y otro con la actualización de aquellos campos en los que vamos a basar la creación del nombre.

Se usará esta combinación:
[Tipo de cita] [Nombre del Paciente] - [Fecha y hora]</td></tr>
<tr><td>Componentes de la solución</td><td>Las características de los flujos de nube a crear serán las siguientes:

<table>
<tr><th>Nombre</th><th>Disparador</th></tr>
<tr><td>Cita.Creacion.ActualizarNombre</td><td>Cuando se crea una nueva cita</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td>Cita.Modificacion.ActualizarNombre</td><td>Cuando se modifican los siguientes campos de una cita: mra_tipodecita, scheduledstart (Tipo de cita y hora inicio)</td></tr></table> En este caso, como el tipo de cita no es obligatorio, se hará una comprobación previa dentro del proceso para generar también el nombre usando la siguiente plantilla: [Nombre del Paciente] - [Fecha y hora]

Nombre	S3.003 - Autocompletar campo nombre para los tratamientos
Escenario	**Como** aplicación de gestión clínica **quiero** que el campo nombre del tratamiento se rellene automáticamente **para** evitar que el usuario tenga que hacerlo manualmente
Descripción	El campo nombre servirá para identificar los distintos tratamientos en la página de vista principal. Con el objetivo de liberar al usuario de la carga de rellenar el nombre manualmente cada vez, dispondremos de un proceso automatizado que haga ese trabajo por el usuario, utilizando una combinación de otros campos (obligatorios) para generar el nombre.
Criterios de Aceptación	**Dado que** el campo nombre está vacío por defecto **cuando** se cree un nuevo tratamiento

<table>
<tr><td></td><td>entonces se ejecutará un proceso que rellene este campo automáticamente
y
Dado que algunos campos obligatorios de los que se extraerá el nombre son editables
cuando se edite uno de esos campos en el tratamiento
entonces se ejecutará un proceso que actualice el nombre automáticamente</td></tr>
<tr><td>Diseño técnico</td><td>Para cumplir el objetivo de este requisito vamos a crear dos flujos de nube, uno que se lanzará con la creación de nuevos tratamientos y otro con la actualización de aquellos campos en los que vamos a basar la creación del nombre.
Se usará esta combinación:
[Tipo de tratamiento] - [Nombre del Paciente]</td></tr>
<tr><td>Componentes de la solución</td><td>Las características de los flujos de nube a crear serán las siguientes:

<table><tr><th>Nombre</th><th>Disparador</th></tr><tr><td>Tratamiento.Creacion.ActualizarNombre</td><td>Cuando se crea un nuevo tratamiento</td></tr><tr><td>Tratamiento.Modificacion.ActualizarNombre</td><td>Cuando se modifican los siguientes campos del tratamiento:
mra_tipodetratamiento</td></tr></table></td></tr>
</table>

Nombre	S3.004 - Autocompletar campo nombre para los diagnósticos
Escenario	**Como** aplicación de gestión clínica **quiero** que el campo nombre del diagnóstico se rellene automáticamente **para** evitar que el usuario tenga que hacerlo manualmente
Descripción	El campo nombre servirá para identificar los distintos diagnósticos en la página de vista principal. Con el objetivo de liberar al usuario de la carga de rellenar el nombre manualmente cada vez, dispondremos de un proceso automatizado que haga ese trabajo por el usuario, utilizando una combinación de otros campos (obligatorios) para generar el nombre.
Criterios de Aceptación	**Dado que** el campo nombre está vacío por defecto **cuando** se cree un nuevo diagnóstico **entonces** se ejecutará un proceso que rellene este campo automáticamente **y** **Dado que** algunos campos obligatorios de los que se extraerá el nombre son editables **cuando** se edite uno de esos campos en el diagnóstico **entonces** se ejecutará un proceso que actualice el nombre automáticamente
Diseño técnico	Para cumplir el objetivo de este requisito vamos a crear dos flujos de nube, uno que se lanzará con la creación de nuevos diagnósticos y otro con la actualización de aquellos campos en los que vamos a basar la creación

<table>
<tr><td></td><td>del nombre.
Se usará esta combinación:
[Tipo de diagnóstico] - [Nombre del Paciente]</td></tr>
<tr><td>Componentes de la solución</td><td>Las características de los flujos de nube a crear serán las siguientes:
<table><tr><th>Nombre</th><th>Disparador</th></tr><tr><td>Diagnostico.Creacion.ActualizarNombre</td><td>Cuando se crea un nuevo diagnóstico</td></tr><tr><td>Diagnostico.Modificacion.ActualizarNombre</td><td>Cuando se modifican los siguientes campos del diagnóstico:
mra_tipodediagnostico</td></tr></table></td></tr>
</table>

Nombre	S3.005 - Bloquear campo cita en la entidad Tratamientos
Escenario	**Como** aplicación de gestión clínica **quiero** bloquear el campo cita en el formulario de tratamiento **para** evitar confusiones para el usuario
Descripción	Al crear un nuevo tratamiento desde el formulario de citas se abre el de tratamiento con el campo cita ya relleno; queremos bloquear ese campo para que el usuario no pueda editarlo.
Criterios de Aceptación	**Dado que** el campo cita ya está relleno, al crear un nuevo tratamiento desde el formulario de citas

	cuando se abra el formulario de creación de tratamientos **entonces** el campo cita estará bloqueado
Diseño técnico	Para conseguir esta lógica en nuestro formulario podemos usar un script usando javascript o crear una nueva regla de negocio para la entidad tratamiento. A la hora de tomar la decisión entre estos dos componentes, ya que en muchos casos son capaces de cumplir las mismas funcionalidades, la prioridad es siempre usar los componentes provistos por Power Apps, ya que estos son más fáciles de mantener a futuro no habría la necesidad de usar ningún otro control de versiones ya que el sistema ya tiene uno incorporado.
Componentes de la solución	Vamos a crear la siguiente regla de negocio: ● Cita fijada y paciente por determinar - bloquear cita Donde primero tendremos una condición que compruebe que el campo cita tiene datos y el campo paciente es nulo y después de esa comprobación haga la acción de bloquear el campo cita.

Nombre	S3.006 - Bloquear campo cita y paciente en la entidad Tratamientos
Escenario	**Como** aplicación de gestión clínica **quiero** bloquear el campo cita y paciente en el formulario de tratamiento **para** evitar que se puedan editar datos ya creados

Descripción	Al editar un tratamiento los campos cita y paciente deberían ser solo de lectura para asegurar la consistencia de los datos después de su creación.
Criterios de Aceptación	**Dado que** los campos cita y paciente ya fueron seleccionados en la creación del tratamiento **cuando** se abra el formulario de edición del tratamiento **entonces** el campo cita y paciente estarán bloqueados
Diseño técnico	Para conseguir esta lógica en nuestro formulario podemos usar un script usando javascript o crear una nueva regla de negocio para la entidad tratamiento. A la hora de tomar la decisión entre estos dos componentes, ya que en muchos casos son capaces de cumplir las mismas funcionalidades, la prioridad es siempre usar los componentes provistos por Power Apps, ya que estos son más fáciles de mantener a futuro no habría la necesidad de usar ningún otro control de versiones ya que el sistema ya tiene uno incorporado.
Componentes de la solución	Vamos a crear la siguiente regla de negocio: ● Paciente y cita configurados evitar edición Donde primero tendremos una condición que compruebe que el campo cita y el campo paciente tienen datos y además el tratamiento tenga nombre, ya que solo puede tener nombre después de haber sido creado, eso nos asegurará que aplicamos la lógica en el formulario de edición. Después de esa comprobación se realizará dos acciones: cambia el campo de cita a modo solo lectura y cambia el campo paciente a modo solo lectura.

Nombre	S3.007 - Bloquear campo cita en la entidad Diagnósticos
Escenario	**Como** aplicación de gestión clínica **quiero** bloquear el campo cita en el formulario de diagnóstico **para** evitar confusiones para el usuario
Descripción	Al crear un nuevo diagnóstico desde el formulario de cita se abre el de diagnóstico con el campo cita ya relleno; queremos bloquear ese campo para que el usuario no pueda editarlo.
Criterios de Aceptación	**Dado que** el campo cita ya está relleno, al crear un nuevo diagnóstico desde el formulario de citas **cuando** se abra el formulario de creación de diagnóstico **entonces** el campo cita estará bloqueado
Diseño técnico	Para conseguir esta lógica en nuestro formulario podemos usar un script usando javascript o crear una nueva regla de negocio para la entidad diagnóstico. A la hora de tomar la decisión entre estos dos componentes, ya que en muchos casos son capaces de cumplir las mismas funcionalidades, la prioridad es siempre usar los componentes provistos por Power Apps, ya que estos son más fáciles de mantener a futuro no habría la necesidad de usar ningún otro control de versiones ya que el sistema ya tiene uno incorporado.
Componentes de la solución	Vamos a crear la siguiente regla de negocio en la entidad diagnósticos: ● Cita fijada y paciente por determinar- bloquear cita

	Donde primero tendremos una condición que compruebe que el campo cita tiene datos y el campo paciente es nulo y después de esa comprobación haga la acción de bloquear el campo cita.

Nombre	S3.008 - Bloquear campo cita y paciente en la entidad Diagnósticos
Escenario	**Como** aplicación de gestión clínica **quiero** bloquear el campo cita y paciente en el formulario de diagnósticos **para** evitar que se puedan editar datos ya creados
Descripción	Al editar un diagnóstico los campos cita y paciente deberían ser solo de lectura para asegurar la consistencia de los datos después de su creación.
Criterios de Aceptación	**Dado que** los campos cita y paciente ya fueron seleccionados en la creación del diagnóstico **cuando** se abra el formulario de edición del diagnóstico **entonces** el campo cita y paciente estarán bloqueados
Diseño técnico	Para conseguir esta lógica en nuestro formulario podemos usar un script usando javascript o crear una nueva regla de negocio para la entidad diagnóstico. A la hora de tomar la decisión entre estos dos componentes, ya que en muchos casos son capaces de cumplir las mismas funcionalidades, la prioridad es siempre usar los componentes provistos por Power Apps, ya que estos son más fáciles de mantener a futuro no habría la necesidad de usar ningún otro control de

	versiones ya que el sistema ya tiene uno incorporado.
Componentes de la solución	Vamos a crear la siguiente regla de negocio en la entidad diagnóstico: ● Paciente y cita configurados - evitar edición Donde primero tendremos una condición que compruebe que el campo cita y el campo paciente tienen datos y además el diagnóstico tenga nombre, ya que solo puede tener nombre después de haber sido creado, eso nos asegurará que aplicamos la lógica en el formulario de edición. Después de esa comprobación se realizará dos acciones: cambia el campo de cita a modo solo lectura y cambia el campo paciente a modo solo lectura.

Nombre	S3.009 - Método de contacto preferente obligatorio
Escenario	**Como** aplicación de gestión clínica **quiero** cambiar la obligatoriedad de los campos de contacto según el método preferente de contacto seleccionado **para** evitar asegurar que se disponen de todos los datos de contacto necesarios para el paciente
Descripción	Cuando el usuario seleccione un método de contacto preferente en el formulario de paciente, el método de contacto seleccionado se volverá obligatorio para el formulario de forma dinámica, de manera que nos aseguramos de que el usuario aporte todos los datos necesarios.

Criterios de Aceptación	**Dado que** el usuario está editando los datos de un paciente **cuando** se seleccione un método de contacto preferente en concreto **entonces** ese método de contacto será obligatorio en el formulario
Diseño técnico	Para conseguir esta lógica en nuestro formulario podemos usar un script usando javascript o crear una nueva regla de negocio para la entidad paciente. A la hora de tomar la decisión entre estos dos componentes, ya que en muchos casos son capaces de cumplir las mismas funcionalidades, la prioridad es siempre usar los componentes provistos por Power Apps, ya que estos son más fáciles de mantener a futuro no habría la necesidad de usar ningún otro control de versiones ya que el sistema ya tiene uno incorporado.
Componentes de la solución	Vamos a crear la siguiente regla de negocio en la entidad paciente: ● El método de contacto preferente seleccionado es obligatorio Y funcionará realizando las siguientes comprobaciones y acciones: ● Primero, comprobará si el método seleccionado es email, en cuyo caso hará obligatorio el campo email y opcionales los demás. ● Segundo, sabiendo que email no se ha seleccionado, comprobará si el método seleccionado es teléfono móvil, en cuyo caso hará obligatorio el campo teléfono móvil y opcionales

	los demás. • Tercero, sabiendo que no es email o teléfono móvil, comprobará si se ha seleccionado la opción teléfono fijo, y de ser así volverá obligatorio este campo y opcionales los demás. • Finalmente, si no es ninguno de los tres, cambiaremos todos los campos a opcionales.

Nombre	S3.010 - Automatización del estado de las citas
Escenario	**Como** aplicación de gestión clínica **quiero** cambiar el estado de las citas cuya fecha y hora de fin esté en el pasado **para** evitar que el usuario tenga que cambiar el estado de las citas manualmente
Descripción	Para que el usuario no tenga que cambiar el estado de las citas de “próxima” a “pasada” de forma manual, el sistema tendrá un proceso automatizado que revisará el estado de las citas próximas todos los días por la madrugada. Todas aquellas citas próximas cuya fecha se encuentre ya en el pasado serán actualizadas, poniendo su estado a “pasada”.
Criterios de Aceptación	**Dado que** una cita que tiene de estado “próxima” ya haya sucedido **cuando** el flujo en la nube se ejecute a las 12:00am **entonces** actualizará su estado a “pasada”
Diseño técnico	Usaremos un flujo en la nube con un disparador recurrente que configuramos para que ejecute las

<table>
<tr><td></td><td>acciones diariamente sobre las doce de la madrugada, para así evitar posibles solapamientos con las acciones del usuario y además distribuir la carga del sistema en horas donde la carga es menor.</td></tr>
<tr><td>Componentes de la solución</td><td>Crearemos el siguiente flujo en la nube:
<table><tr><th>Nombre</th><th>Disparador</th><th>Filtros</th></tr><tr><td>Cita.CadaDia.
RevisarEstado</td><td>Cada día a las 12:00am.</td><td>Filtro de datos:
scheduledend,
statecode,
activityid

Filtro logico:
mra_hoy gt
scheduledend
and statecode eq
0</td></tr></table></td></tr>
</table>

Nombre	S3.011 - Campo Documento de autorización en la entidad tratamiento solo es visible si es requerido
Escenario	**Como** aplicación de gestión clínica **quiero** hacer que el campo Documento de autorización solo sea visible para el usuario si este es requerido **para** solo mostrarlo si fuera necesario
Descripción	Cuando el usuario seleccione si el tratamiento tiene autorización entonces el campo Documento de autorización se hará visible y será obligatorio.

Criterios de Aceptación	**Dado que** el usuario está editando los datos de un tratamiento **cuando** se seleccione que el tratamiento requiere autorización **entonces** se mostrará el campo Documento de autorización y será un campo obligatorio.
Diseño técnico	En este caso no es posible usar una regla de negocio para operar sobre el campo Documento de autorización, así que nos veremos obligados a crear un script que funcionará sobre el formulario de la entidad tratamiento.
Componentes de la solución	Crearemos un nuevo recurso web en nuestra solución donde guardar todos los scripts de la entidad Tratamiento que se llamará tratamientos.js Dentro de ese archivo introduciremos todas las funciones que necesitemos para después configurarlas en el formulario donde corresponda.

Nombre	S3.012 - Aviso de que la fecha sea futura en la creación de una nueva cita
Escenario	**Como** aplicación de gestión clínica **quiero** que salte un aviso cuando la fecha de inicio de una cita se encuentre en el pasado **para** avisar al usuario de que la fecha no es correcta y pueda corregirla
Descripción	Cuando el usuario se encuentre creando una nueva cita y este seleccione la hora de inicio de la cita, el guardar los datos, el sistema evaluará si la fecha es correcta y en el

	que caso de que no lo sea dará un aviso, recomendando al usuario que introduzca una fecha futura.
Criterios de Aceptación	**Dado que** el usuario está creando/editando los datos de una cita **cuando** se seleccione la hora de inicio **entonces** si esa fecha y hora no fueran correctas, un mensaje de alerta saltaría al intentar guardar los datos
Diseño técnico	Aquí volveremos a usar una regla de negocio con la que daremos una alerta para el usuario, con un mensaje en pantalla diciendo que seleccione una fecha y hora en el futuro. Para esta comprobación se tendrá en cuenta tanto la fecha como la hora de la cita.
Componentes de la solución	Para que el sistema tenga registrado la fecha y hora de hoy para hacer las comprobaciones, crearemos un nuevo campo en la entidad cita que se llamará “Hoy” y contendrá la fecha y hora de hoy. Vamos a crear la siguiente regla de negocio en la entidad cita: ● Aviso Hora inicio futura Esta usará el nuevo campo “Hoy” para compararlo con la Hora de inicio de la cita y en caso de la hora de inicio sea menor a hoy, se lanzará un mensaje de error que impedirá que se guarden los datos hasta que se introduzca una fecha válida, con el mensaje: Por favor introduzca una fecha futura.

4.4.2 Evaluación del desarrollo del tercer sprint

Además de los requisitos mencionados anteriormente también se editó el campo "estado" de la entidad citas, para reducir el número de elecciones disponibles para el usuario. Para ello creamos una solución auxiliar sólo con la entidad Cita y el campo "estado", y la exportamos. En la Figura 19 se muestra la estructura base del fichero de solución en formato XML. Si navegamos dentro de esa estructura, llegaremos a la parte que contiene los datos sobre los campos de la entidad, tal y como vemos en la Figura 20, editando el fichero y volviendo a importar la solución pudimos editar este campo que a priori no es editable, por pertenecer a una entidad OOB.

Figura 19. Estructura del fichero de solución

Figura 20. Vista de los estados dentro del fichero de configuración

Si hablamos de los elementos que han sido introducidos en la solución principal dentro de esta iteración tenemos un total de:

- Siete flujos automatizados en la nube, uno de ellos programado.
- Seis reglas de negocio repartidas entre las distintas entidades principales: citas, tratamientos, diagnósticos y pacientes.
- Una nueva entidad, Tipos de Procedimientos
- Un script para el formulario de la entidad tratamientos.

Empezaremos hablando de la nueva entidad descrita en el requisito *S3.001*, Tipo de Procedimiento, que hemos introducido en esta iteración, con el objetivo de añadir aún más funcionalidad automatizada a los formularios se ha necesitado la creación de esta nueva estructura de datos que sustituye los anteriores campos de "tipo de tratamiento" y "tipo de diagnóstico" para la entidad tratamiento y diagnóstico respectivamente.

La funcionalidad específica que cubre esta nueva entidad es la de recopilar los datos comunes de tratamientos y diagnósticos, teniendo así un registro de datos accesible, personalizable y reutilizable para el usuario. Para ello se han creado todos los elementos necesarios para la nueva entidad, incluyendo también a esta en el menú principal de nuestra aplicación.

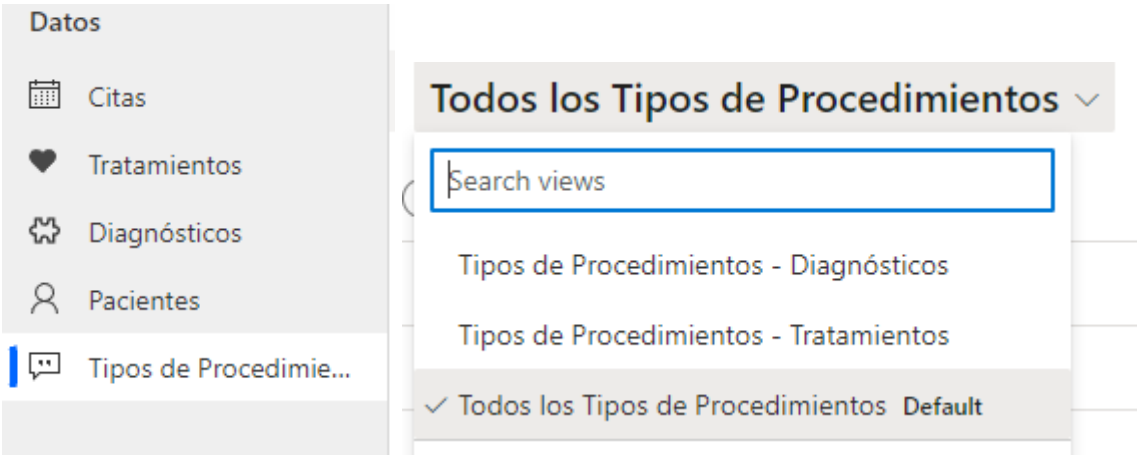

Figura 21. Menú lateral y vistas

Dispone de cuatro nuevas vistas, aunque en la interfaz, como se ve en la Figura 21, solo veremos tres, ya que una de ellas es simplemente el componente que nos permite usar el buscador, y no se muestra como vista principal, se usa para filtrar por los campos de la entidad. Las vistas “Tipos de Procedimientos - Diagnósticos” y “Tipos de Procedimientos - Tratamientos” se usan tanto en la página principal de la entidad como en los formularios de diagnósticos y tratamientos para filtrar los tipos de procedimientos como corresponda, mostrando solo los tipos de procedimientos de tipo diagnóstico en el formulario de diagnóstico y lo mismo para los tratamientos, podemos ver un ejemplo en la Figura 22.

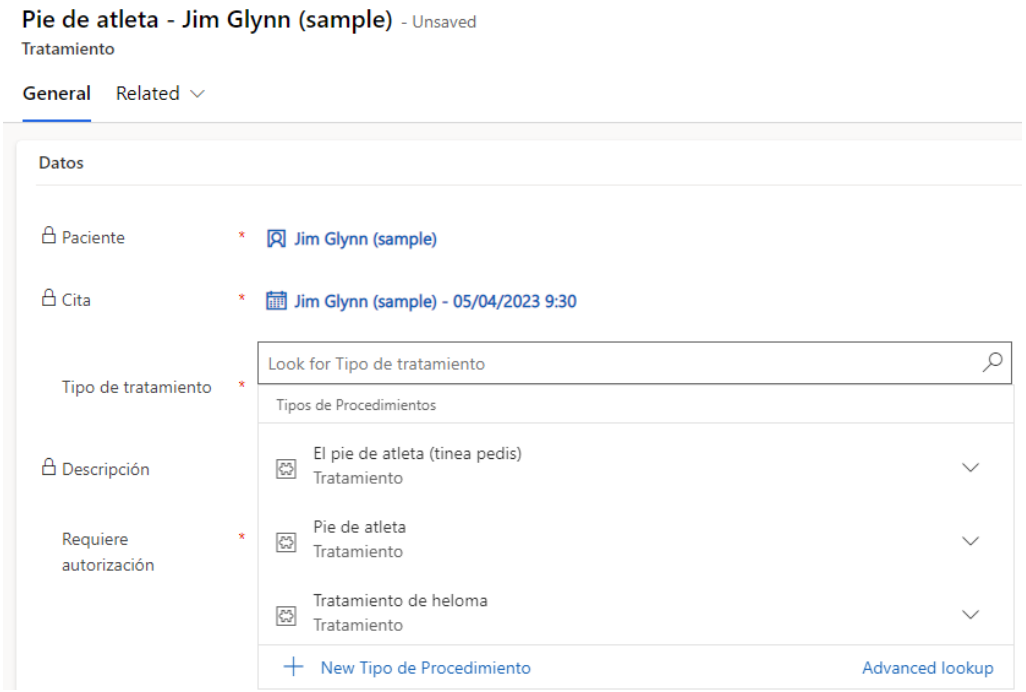

Figura 22. Selección de tipo de tratamiento

El siguiente requisito de este sprint, *S3.002*, forma parte de una serie de requisitos que consiste en la autocompletación de los nombres de las entidades como cita, en este caso, y también tratamientos o diagnósticos. Vamos a ver el caso específico de la entidad cita, para ello se han creado dos flujos en nube de Power Automate, los cuales funcionan en creación de la cita y en modificación de alguno de sus campos,

de tal manera que el nombre es calculado o recalculado para facilidad del usuario y una buena identificación de los datos una vez ubicados en las vistas.

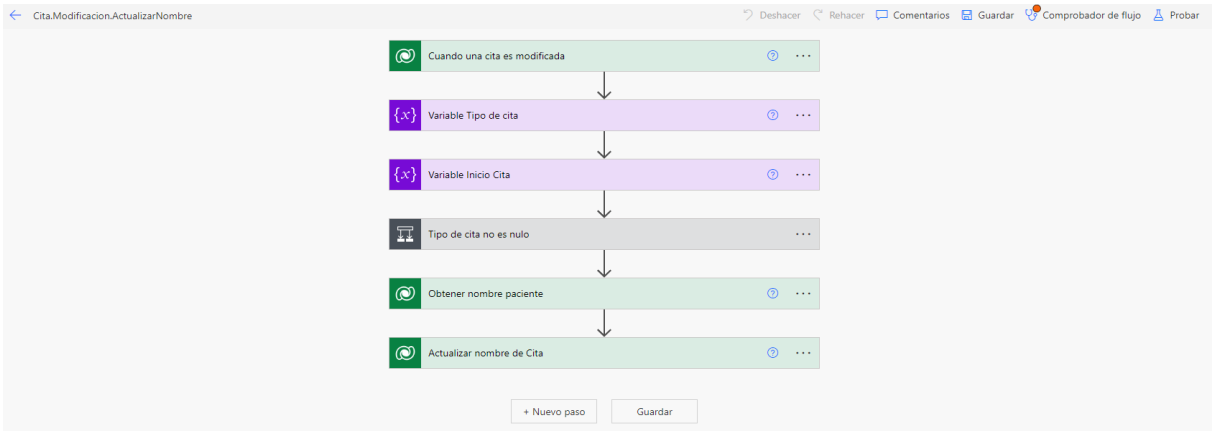

Figura 23. Vista general del flujo en la nube

Como se puede observar en la Figura 23, hemos usado variables para almacenar los datos con los que hemos compuesto el nombre. Usar variables nos facilita el dar forma a los datos, una de las problemáticas que se encontraron en el desarrollo fue traducir los valores lógicos del tipo de cita a valores de texto y dar formato a la fecha y hora de la cita.

Name	Value
Consulta	866040000
Tratamiento	866040001
Seguimiento	866040002

Figura 24. Tabla de valores del tipo de cita

Para ello hemos usado un componente conmutador para traducir el valor numérico del tipo de cita a su valor textual y, en el caso del formato de la fecha, una fórmula. Podemos ver la tabla de valores en la Figura 24 y la fórmula para la fecha en la Figura 25.

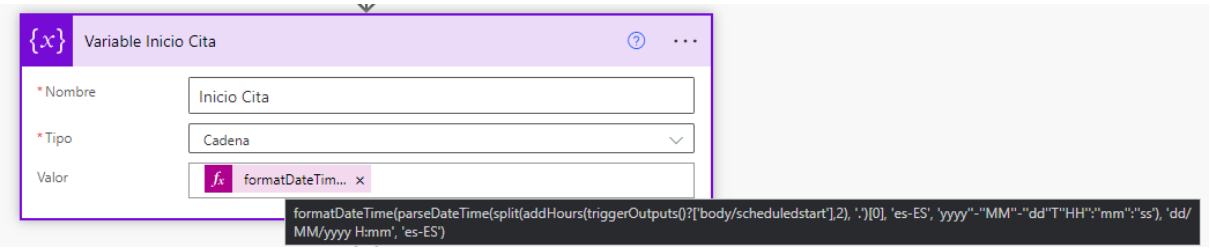

Figura 25. Fórmula de formato de fecha

El resto de flujos en la nube desarrollados en esta iteración son similares, salvo el construido para el requisito *S3.010*. Este flujo tiene la particularidad de funcionar de manera periódica y programada, todos los días a una hora concreta, además de recopilar una serie de datos, usando un filtro concreto y realizar operaciones sobre todos ellos, usando el componente de bucle.

Lo primero que vemos en la Figura 26 es el lanzador del flujo. Como podemos observar, está configurado para lanzarse diariamente. Después de la acción inicial del flujo tenemos una de Dataverse donde lanzamos una petición de datos, observamos un filtro de datos y una expresión lógica. Esta acción nos devolverá una lista de citas, y finalmente vemos el elemento de bucle que se ejecutará una vez por cada dato de la lista anterior, realizando la acción de tipo Dataverse que actualizará el estado de la cita en cuestión.

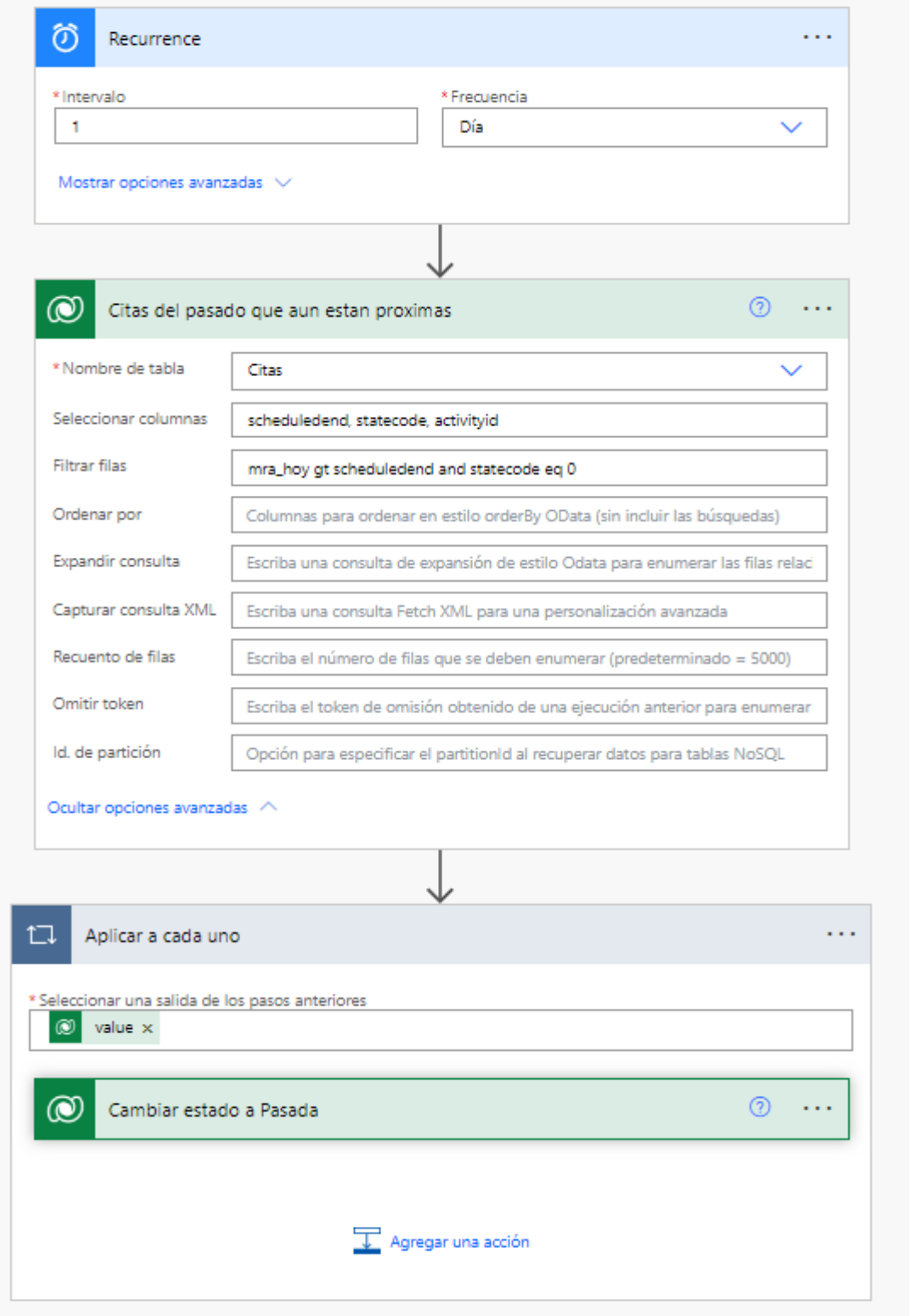

Figura 26. Flujo de nube recurrente

Lo siguiente que veremos como ejemplo de los componentes creados en esta iteración será una de las reglas de negocio, en concreto la perteneciente al requisito *S3.009*. Para este requisito modificamos de forma dinámica la obligatoriedad de los

campos en la sección “datos de contacto” del formulario de paciente, que se muestran en la Figura 27.

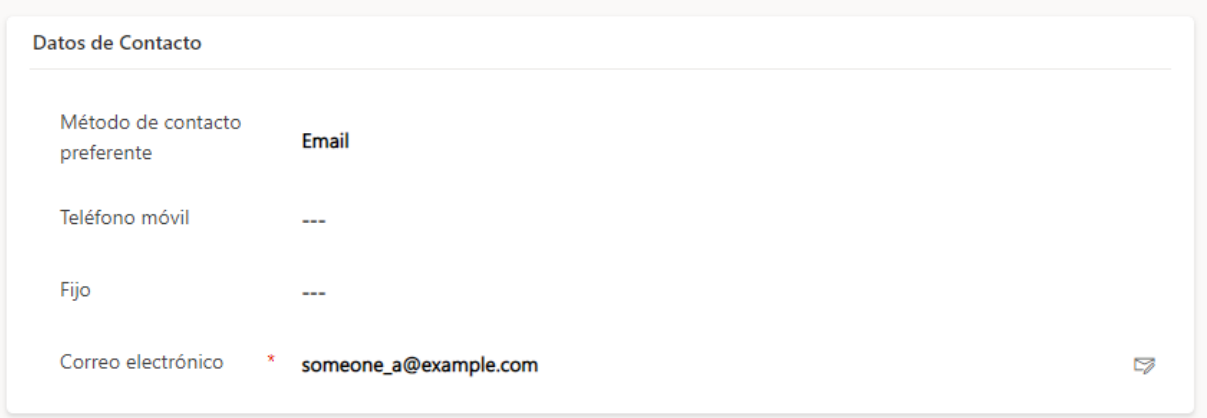

Figura 27. Sección “Datos de Contacto” en el formulario de Paciente

Para la lógica hemos aplicado una serie de comprobaciones en cadena en las que, en cada una, realizamos las acciones necesarias para añadir las restricciones de obligatoriedad pertinentes a cada campo de esta sección, como vemos en la Figura 28.

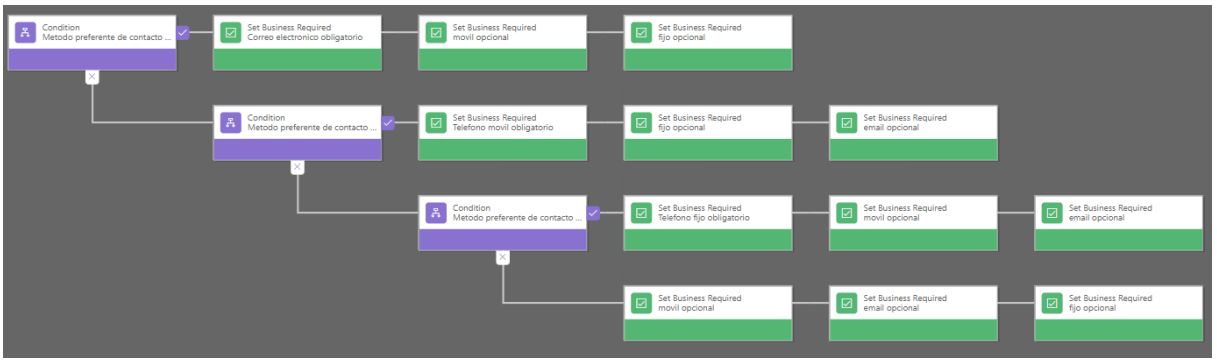

Figura 28. Regla de negocio para el formulario de Paciente

A continuación, ya habiendo visto como el uso de las reglas de negocio nos permiten editar la lógica de los formularios, pasamos a exponer uno de los ejemplos donde el uso de las reglas de negocio era insuficiente (debido al tipo de datos del campo), *S3.011*.

En este requisito hemos necesitado crear un recurso web dentro de nuestra solución, como muestra la Figura 29, para almacenar el código javascript que contiene la lógica concreta que deseamos aplicar al formulario.

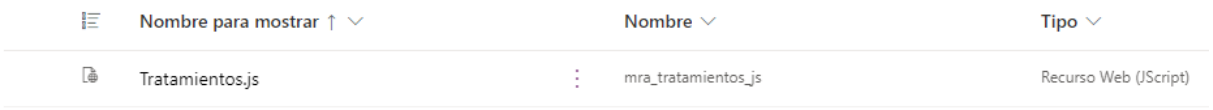

Figura 29. Recurso web para el script de la entidad tratamientos

La Figura 30 muestra la sección “Biblioteca de formularios”, donde se registran los recursos web como librerías. Para usar el recurso en el formulario tenemos que registrarlo y además suscribir las funciones que vayamos a utilizar donde sea necesario.

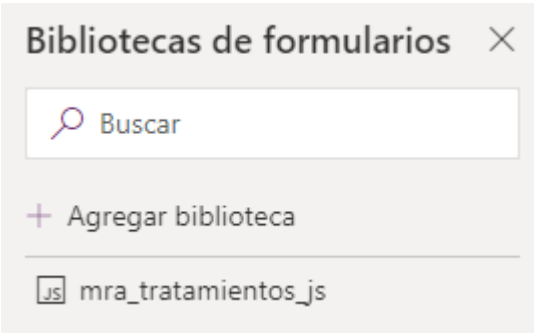

Figura 30. Recurso web añadido a la lista de librerías del formulario

En la Figura 31 vemos qué funciones hemos registrado en cada evento del formulario. En este caso hemos suscrito la función principal de nuestro script al evento “Al cargar” del formulario. Esto es parte de unas buenas prácticas que, de ser necesario, nos permitirá registrar nuevas funciones a este evento del formulario sin tener que añadirlas de nuevo, solo editando la parte correspondiente de nuestro script. Y para este caso concreto también hemos registrado la función en el evento “Al cambiar” del campo “Requiere autorización”, lo que significa que nuestra función se ejecutará además cuando tenga lugar un cambio en el campo.

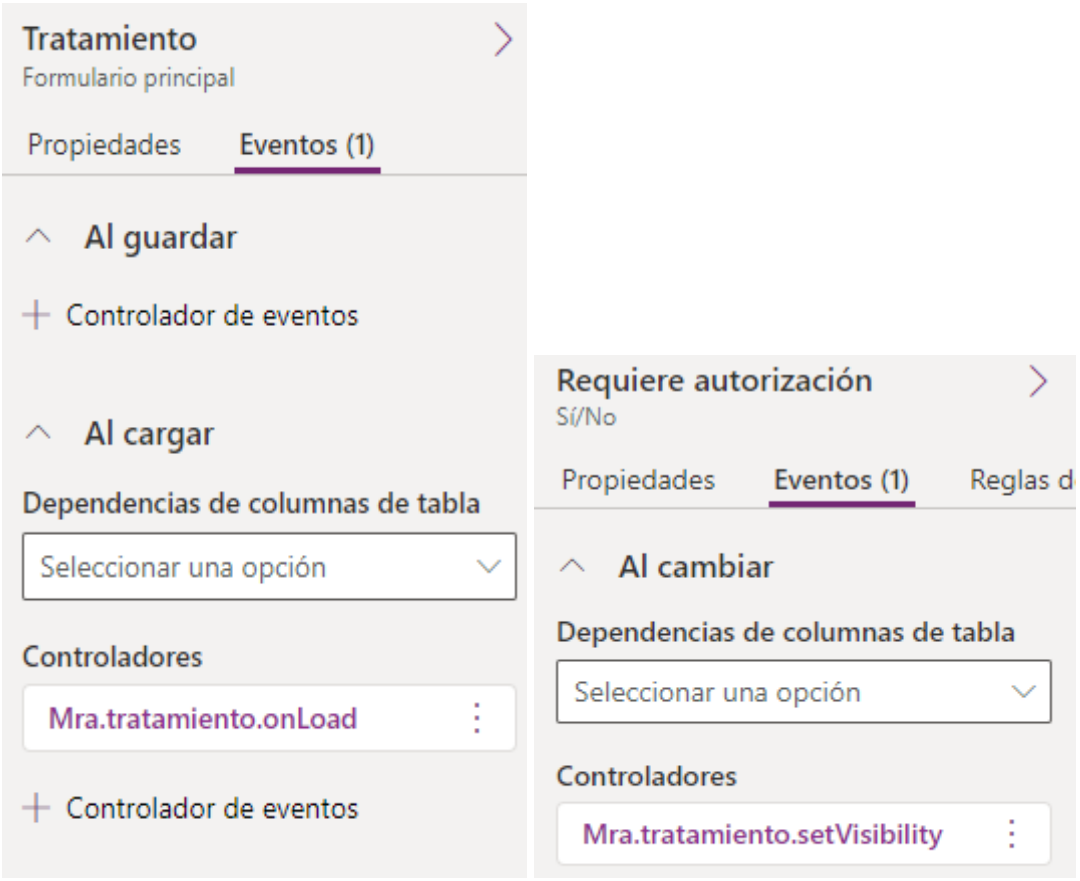

Figura 31. Funciones registradas en los eventos del formulario de tratamientos

La Figura 32 muestra el código desarrollado, que utiliza una estructura concreta, usando el formato de espacios de nombre recomendado por Microsoft [25].

```
JS tratamientos.js > ...
1   var Mra = window.Mra || {};
2   Mra.tratamiento = Mra.tratamiento || {};
3
4   (function(){
5
6   //funcion de onLoad
7   this.onLoad = function (executionContext){
8       Mra.tratamiento.setVisibility(executionContext);
9   }
10
11  this.setVisibility = function(executionContext){
12      debugger;
13      //get formcontext
14      var formContext= executionContext.getFormContext();
15      //get object data
16      var isRequired = formContext.getAttribute("mra_isauthorizationrequired").getValue();
17      formContext.getControl("mra_autorizacion").setVisible(isRequired);
18      if(isRequired){
19          formContext.getAttribute("mra_autorizacion").setRequiredLevel("required");
20      }else{
21          formContext.getAttribute("mra_autorizacion").setRequiredLevel("none");
22      }
23
24  }
25
26  }).call(Mra.tratamiento);
```

Figura 32. Código javascript para el requisito *S3.011*

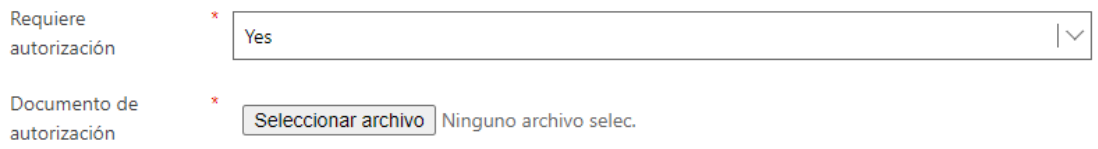

Figura 33. Campo Documento de autorización

Finalmente, en la Figura 33, podemos observar que el campo “documento de autorización” se hace visible cuando la autorización es requerida, e invisible cuando la autorización no es requerida. Además al mismo tiempo que se hace visible también se hace obligatorio en el formulario, lo que obliga al usuario a incluir el documento de autorización antes de poder guardar los datos.

4.5 Desarrollo del cuarto sprint

En esta iteración vamos a cubrir los componentes y funciones más complejas de nuestro sistema. Nos vamos a centrar en dos funcionalidades concretas, la recogida de firma electrónica y el aviso por correo electrónico. Para la primera necesitaremos crear una aplicación complementaria basada en dispositivos móviles y tabletas que nos dé una funcionalidad básica de visualización de los datos además de la capacidad de recoger las firmas de los pacientes, almacenándose en el tratamiento que corresponda. El aviso por correo electrónico se realizará usando componentes que ya hemos visto en la iteración anterior, que son los flujos en la nube de Power Automate.

4.5.1 Requisitos para el cuarto sprint

Nombre	S4.001 - Creación de nueva aplicación para tabletas con funcionalidad para la recogida de firmas digitales
Escenario	**Como** usuario de la aplicación **quiero** poder usar un método electrónico de recogida de firmas para el consentimiento de los tratamientos **para** evitar tener que escanear y usar documentos en papel
Descripción	Crearemos una nueva canvas app que conecte con los datos que ya usamos en Dataverse para que nos permita usar una tablet o un móvil para recoger la firma digital de los pacientes.
Criterios de Aceptación	**Dado que** el usuario quiere acceder a la nueva aplicación en su dispositivo móvil o tablet **cuando** se descargue la aplicación Power Apps **entonces** al registrarse con su usuario y contraseña encontrará la aplicación de firma electrónica
Diseño técnico	Crearemos una nueva aplicación de lienzo con interfaz de tableta, con el objetivo de integrar la funcionalidad necesaria para que el usuario pueda recoger la firma digital de los pacientes en el caso de que reciban un tratamiento en la clínica que requiera de autorización firmada.
Componentes de la solución	Llamaremos a nuestra nueva app “App Firma digital”

Nombre	S4.002 - Interfaz de selección del tratamiento para el cual se recogerá la firma
Escenario	**Como** usuario de la aplicación **quiero** poder visualizar los datos en la aplicación de firma digital **para** seleccionar el tratamiento para el cual se va a recoger la firma
Descripción	El usuario, al entrar a la aplicación de firma digital, además de poder recoger la firma digital de un paciente, debe ser capaz de seleccionar para qué tratamiento será esa firma.
Criterios de Aceptación	**Dado que** el usuario quiere recoger una firma para un tratamiento **cuando** inicie la aplicación de firma digital **entonces** podrá ver los registros de los tratamientos que ya están registrados en el sistema
Diseño técnico	Para hacer que la visualización de los datos sea más intuitiva se plantea realizar el proceso de visualización y selección de datos en dos pasos: primero se visualizarán todos los pacientes del sistema y se seleccionará uno, después veremos una lista de los tratamientos creados para el paciente seleccionado.
Componentes de la solución	Dentro de la aplicación crearemos dos pantallas, por defecto, en la parte superior de cada pantalla, tendremos un título de navegación con el nombre de la aplicación y una breve descripción del objetivo o las acciones que se pueden realizar en la página.

	Los títulos tendrán el siguiente formato: ● Pantalla 1, “Firma digital - Pacientes” ● Pantalla 2, “Firma digital - Tratamientos” En cada pantalla usaremos el elemento galería para mostrar una vista de los datos tanto de pacientes como de tratamientos. Para los pacientes mostraremos los campos nombre completo, profesión y su foto; y para los tratamientos mostraremos el nombre, nombre de la cita a la que está asociado y descripción. Además añadiremos un par de elementos de navegación en la segunda pantalla para que el usuario pueda volver a la pantalla de inicio.

Nombre	S4.003 - Recogida de firma para el tratamiento seleccionado
Escenario	**Como** aplicación de recogida de firmas **quiero** proveer una pantalla con la funcionalidad de recogida de firma digital **para** que el usuario recoja las firmas de los pacientes
Descripción	Una vez habiendo seleccionado el tratamiento para el cual se quiere recoger la firma, se podrá navegar a una última pantalla donde se mostrará la interfaz de recogida de firma, en la cual el paciente podrá dejar su firma autorizando así el tratamiento.
Criterios de Aceptación	**Dado que** el paciente debe firmar el consentimiento del tratamiento

	cuando el usuario seleccione el tratamiento a firmar **entonces** se mostrará la pantalla de firma digital para recoger la firma del paciente
Diseño técnico	En esta pantalla, además de disponer de los controles necesarios para que el paciente pueda firmar sobre la pantalla de la tableta o dispositivo móvil que esté usando el usuario, se realizarán las acciones necesarias para que la imagen de firma se guarde de forma automática en el registro de tratamiento que corresponda, conectando los datos con Dataverse. De tal manera que cuando el usuario vuelva a consultar el tratamiento desde la aplicación principal pueda ver la firma del paciente.
Componentes de la solución	Usaremos el componente entrada manuscrita para que el usuario dibuje su firma. Dentro del cuadrado blanco de escritura añadiremos una línea de guía sobre la que firmar (esta no aparecerá en la imagen final) y a un lado pondremos un botón de aceptar, que desencadenará las acciones de registro de la firma en el tratamiento que corresponda, vinculando así los datos recogidos con Dataverse. Una vez finalizado el proceso se volverá automáticamente a la pantalla inicial. Además añadiremos un par de elementos de navegación en la para que el usuario pueda volver tanto a la pantalla anterior como a la pantalla de inicio, para que pueda volver a seleccionar el tratamiento en caso de haberse equivocado.

Nombre	S4.004 - Mejora de la visualización de los datos de tratamientos y pacientes en la aplicación de firma digital
Escenario	**Como** aplicación de recogida de firmas **quiero** filtrar los datos de pacientes y tratamientos **para** facilitar al usuario la visualización de los datos
Descripción	Con el objetivo de mejorar la visualización de los datos se añadirá la opción de filtrar a los pacientes según el nombre y además se limitará aún más el conjunto de tratamientos que se mostrarán después de seleccionar el paciente, solo mostrando aquellos que estén pendientes de firmar y la firma sea requerida
Criterios de Aceptación	**Dado que** el conjunto de datos que se muestra es muy grande **cuando** el usuario vea los datos de pacientes **entonces** podrá usar una entrada de texto para filtrar a los pacientes por su nombre **y** **Dado que** el conjunto de datos que se muestra es muy grande **cuando** el usuario vea los datos de tratamientos **entonces** se habrán filtrado previamente para mostrar solo los que estén marcados como pendientes a firmar
Diseño técnico	Tenemos que añadir varios componentes a nuestra aplicación para conseguir los resultados deseados. Lo primero sería conseguir registrar en la aplicación cuando un tratamiento, que es de firma requerida está aún pendiente de firma o no, y una vez teniendo esos datos, los usaremos directamente para aplicar un filtro sobre los

	tratamientos. Además se añadirá un buscador en la página de pacientes, para que así sea posible para el usuario buscar un paciente concreto sin tener que navegar por toda la lista de pacientes hasta encontrarlo manualmente.
Componentes de la solución	Nuevo campo en la entidad tratamientos: Pendiente de firma (mra_pendientedefirma), de tipo Yes/No Para este campo además se creará una nueva regla de negocio que actualice su valor de forma automática, si la firma es requerida y aun no se ha firmado entonces pendiente de firma será igual a “Sí”, en cualquier otro caso será igual a “No”. El nombre de la regla de negocio será el siguiente: ● Pendiente de firma - asignar valor según firmado o no firmado También se cambiará la lógica para permitir al usuario cambiar el campo requiere firma a “Sí” sin aun tener la firma del paciente en el tratamiento, para así detectar cuáles son los tratamientos pendientes de firma. Con respecto al filtro de pacientes, se añadirá un campo de texto en la pantalla principal que conectará con la lista de pacientes, de tal manera que comprobará si el texto introducido por el usuario está contenido en el nombre

	completo de los pacientes y mostrará sólo aquellos que cumplan esta condición. Cuando el campo de búsqueda de texto esté vacío mostrará todos los pacientes del sistema, para limpiar el campo de texto se dispondrá de un botón al lado de este que borrará todo lo que haya escrito el usuario en su última búsqueda.

Nombre	S4.005 - Aviso por email de la próxima cita
Escenario	**Como** aplicación de gestión clínica **quiero** enviar un email al paciente un día antes de su cita **para** recordarle que tiene una cita próxima
Descripción	Para mejorar la asistencia de los clientes de la clínica se propone activar un proceso automático que diariamente compruebe si hay citas al día siguiente, para que dé un aviso al paciente de que tiene una cita próximamente.
Criterios de Aceptación	**Dado que** el paciente tiene una cita próxima **cuando** la fecha de la cita sea mañana **entonces** el sistema avisará al paciente mediante un email de que tiene una cita
Diseño técnico	Usando Power Automate podemos crear un flujo en la nube automatizado que se ejecute diariamente para comprobar si hay citas próximas para el día de mañana, y que además se encargue de mandar un email (si corresponde) al paciente citado. Para asegurar que la comunicación es adecuada, este proceso solo se lanzará para aquellos pacientes cuyo método de contacto preferente sea el correo electrónico.

<table>
<tr>
<td>Componentes de la solución</td>
<td>Nuevo flujo en la nube:

<table>
<tr><th>Nombre</th><th>Disparador</th><th>Filtros</th></tr>
<tr><td>Cita.Diariamente.AvisarPorMetodoContactoPreferente</td><td>Cada día a las 10:00am.</td><td>Para el filtro de datos y el filtro lógico se utilizará un fetchXML</td></tr>
</table>

Usaremos FetchXML, [23], para acceder a los datos del lookup Paciente de la entidad de citas sin tener que hacer llamadas adicionales a Dataverse. En este caso vamos a procesar varios registros a la vez (todas las citas que estén próximas, ocurran mañana y el paciente tenga “correo electrónico” como método de contacto preferente), para optimizar los tiempos de respuesta y carga de datos, vamos a evitar consultar los datos cada vez que procesamos una cita, utilizando FetchXML, con una sola consulta traeremos todos los datos que necesitamos.

Para enviar el email, configuraremos una nueva referencia de conexión, creando un usuario específico con el que conectaremos los servicios de correo y será nuestro remitente para todas las notificaciones de citas.</td>
</tr>
</table>

4.5.2 Evaluación del desarrollo del cuarto sprint

Una de las funciones más importantes con respecto a este proyecto la desarrollamos en este sprint. Encargándose de la integración y desarrollo de una nueva aplicación para nuestra solución, que nos permita recoger firmas de los pacientes desde dispositivos móviles o tablets. Esto es de gran importancia, ya que el objetivo es proveer una solución tecnológica que mejorará la productividad de la clínica y para ello es muy importante disponer de componentes que cubren todas las necesidades principales de esta.

Primeramente, creamos una nueva aplicación de lienzo en Power Apps, *S4.001*, la cual después conectaremos con nuestros datos en Dataverse, tal como se ve en la Figura 34. De esta forma logramos comunicar las dos aplicaciones, canvas app y model driven app, ya que ambas estarían conectadas a la misma fuente de datos.

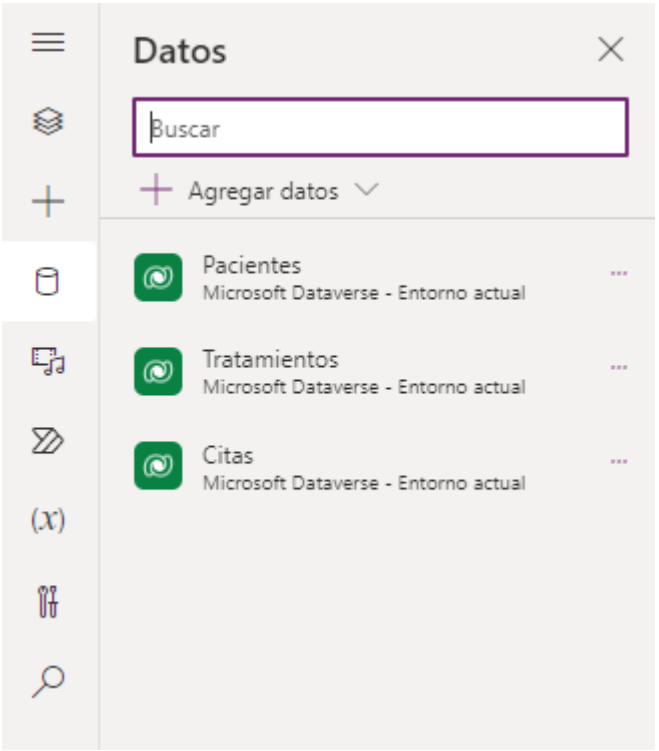

Figura 34. Datos integrados con canvas app desde el menú de la pantalla de edición

El desarrollo de una canvas apps dista mucho de lo que hemos visto con nuestra aplicación principal, por lo que se han encontrado una serie de dificultades, en concreto debido a los cambios de idioma. Como hemos mencionado anteriormente, dentro del entorno de Power Platform existe un lenguaje llamado Power Fx [5] utilizado en todas las funciones de la aplicación de lienzo que hemos construido. Pero debido a opciones de idioma, este lenguaje tiene una sintaxis distinta en inglés que en español. Por ello ha surgido una dificultad a la hora de encadenar varias funciones en la misma sección, debido a que uno de los caracteres no era válido.

Para encontrar la solución a este problema hemos hecho uso de una de las ventajas que tenemos de cara al desarrollo con Power Platform, el foro de la comunidad [4]. Una vez publicado nuestro mensaje explicando el error concreto al que nos estábamos enfrentando, Figura 35, en poco tiempo un usuario nos ha brindado una solución al problema de sintaxis, como se ve en la Figura 36.

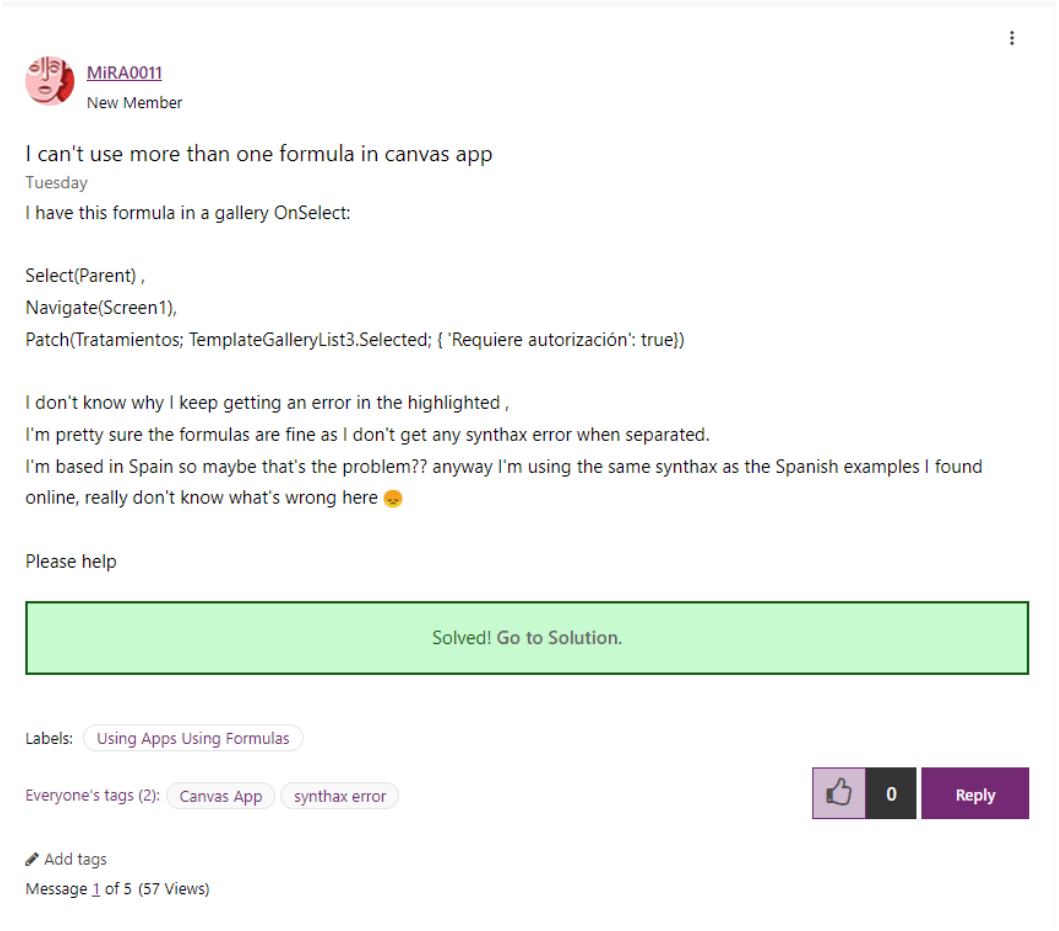

Figura 35. Mensaje publicado en el foro “Get Help with Power Apps”

Figura 36. Respuesta aportada por un usuario del foro

Una vez resueltas las dudas técnicas, nos centramos en construir las pantallas necesarias para la visualización de los datos de pacientes y tratamientos, *S4.002*. Usando los controles de galería que ofrece Power Apps por defecto, qué se conectan a una de las fuentes de datos que tengamos registradas en la aplicación, podemos crear listas de datos rápidamente, en las cuales se facilita incluir y editar campos para su visualización añadiendo etiquetas de texto dentro de este elemento galería y posteriormente, editando la propia galería para que relacione cada etiqueta con uno de los atributos de la fuente de datos que tiene asignada, vemos un ejemplo en la Figura 37.

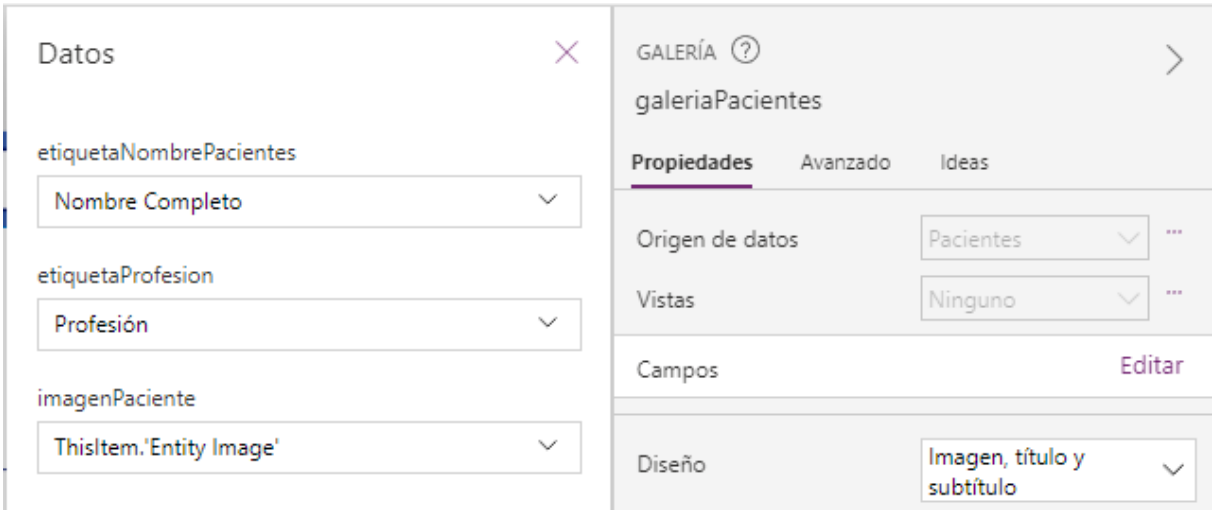

Figura 37. Edición de campos en la galería de pacientes

Finalmente añadimos un campo de búsqueda para filtrar los datos de pacientes, conectamos este campo con la fuente de datos de la galería y obtenemos el resultado final para el requisito *S4.004*, que podemos ver en la Figura 38.

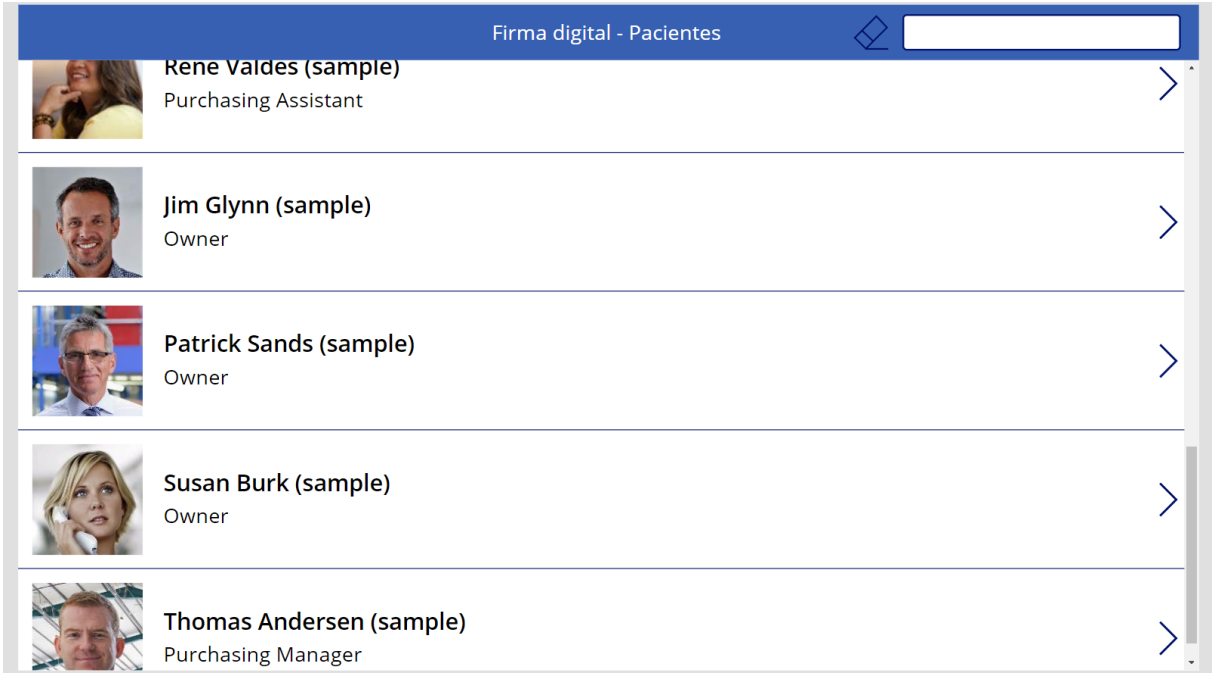

Figura 38. Pantalla de selección de pacientes

Si seleccionamos algún paciente pasaremos a la siguiente pantalla donde podremos ver una lista de tratamientos relacionados con el mismo. Además sólo se mostrarán los tratamientos de ese paciente cuya firma sea requerida y esté

pendiente de firmar. En caso de que al seleccionar un paciente no haya ningún tratamiento que satisfaga estas condiciones, se mostrará un mensaje avisando de que no hay tratamientos que mostrar para ese paciente, véase Figura 39, y el usuario podrá volver atrás desde cualquiera de los dos botones de navegación que se encuentran en la parte superior de la pantalla.

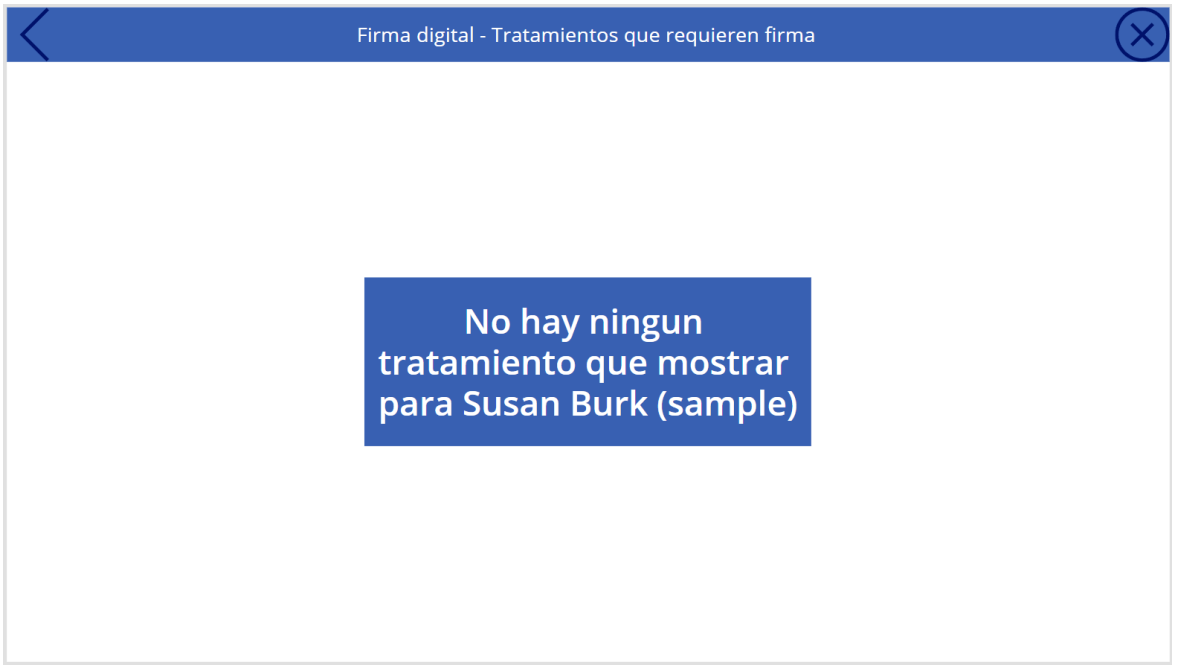

Figura 39. Pantalla de tratamientos cuando no hay tratamientos pendientes de firma

Dentro de la visualización de los tratamientos, Figura 40, podemos ver el nombre del tratamiento, el nombre de la cita y la descripción del tratamiento. Una vez seleccionado el tratamiento concreto, usando el botón de selección, para el que queremos recoger la firma, navegaremos a la siguiente pantalla.

Figura 40. Pantalla de tratamientos

En la siguiente pantalla, *S4.003*, dispondremos de nuevo de dos botones de navegación en la parte superior. En este caso la flecha hacia atrás nos llevará a la sección anterior donde podremos volver a elegir el tratamiento y usando el botón de cancelar nos llevará a la pantalla de inicio, desde la cual podremos empezar el proceso desde el principio eligiendo el paciente. También para facilitar la labor al usuario, el título de esta pantalla de firma contendrá el nombre del tratamiento y del paciente que ha sido seleccionado a lo largo del proceso, verificando así que el registro seleccionado es correcto.

Aparte, el elemento de recogida de firma dispone de una serie de controles por defecto, ya provistos por Power Apps, que nos permitirán borrar la firma u otras ediciones sobre la escritura, podemos ver estos controles en la Figura 41. Finalmente para guardar la imagen de la firma, y que esos datos sean escritos en el registro de Dataverse que corresponda, el usuario deberá pulsar el botón de tick o confirmación blanco que se ve en la parte inferior derecha de la pantalla.

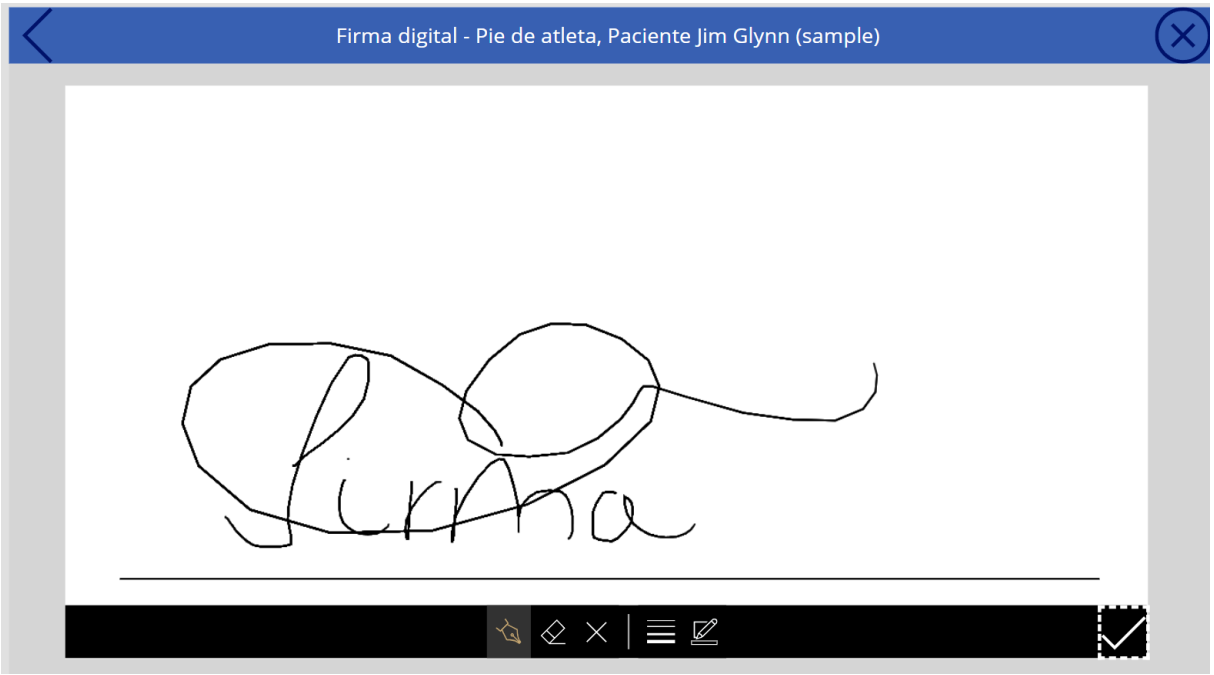

Figura 41. Pantalla de firma

Después de haber concluido con todo el proceso en la aplicación de firma podremos volver a nuestra aplicación principal para ver los resultados. Podremos observar dos cosas, el campo pendiente de firma tiene ahora el valor “No” y abajo de él se encuentra la firma del paciente, tal y como vemos en la Figura 42. Para mejorar la consistencia de los datos, además se ha añadido una lógica al formulario que impida editar la firma una vez haya sido registrada.

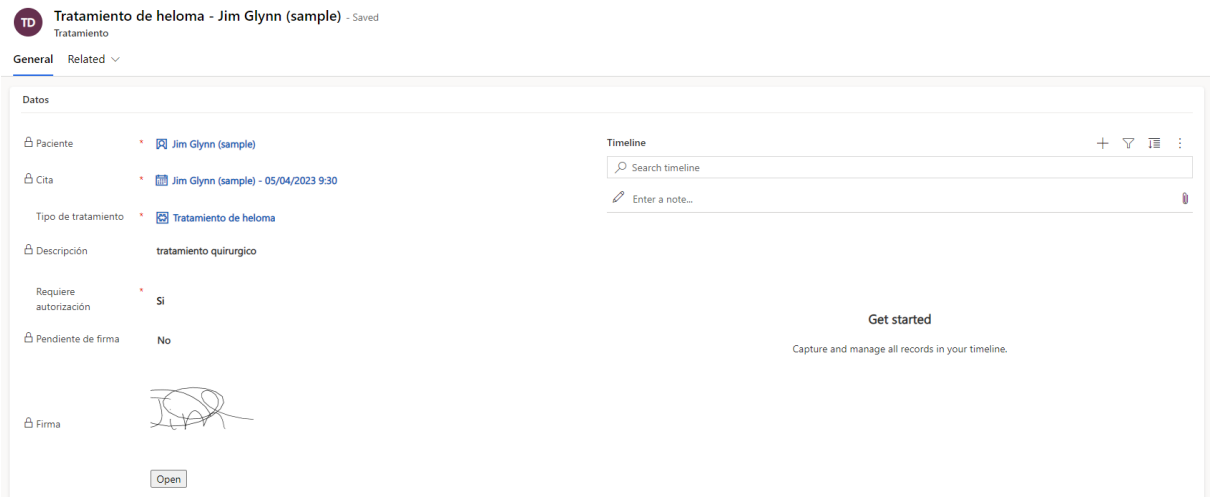

Figura 42. Tratamiento firmado

Finalmente, para concluir con el desarrollo del proyecto, hemos desarrollado un flujo en la nube que se encargue de consultar diariamente las citas, *S4.005*, recopilando las que estén próximas, cuya fecha de inicio sea mañana y que además el paciente asociado a estas citas tenga como método de contacto preferente seleccionado el correo electrónico. Para realizar este filtro tan concreto y además traer todos los datos necesarios para redactar el correo en una sola consulta a Dataverse, optimizando así la cantidad de datos por llamada y el número de llamadas a datos, hemos hecho uso de FetchXML [23].

Para extraer el FetchXML que necesitamos hemos hecho uso de la función de búsqueda avanzada que nos provee nuestra aplicación, Figura 43, en la que podemos, mediante el uso de una interfaz visual, seleccionar o crear la lógica que queremos para nuestro filtro, al igual que seleccionar que campos queremos que se traiga la llamada a datos una vez lo usemos en nuestro flujo automatizado.

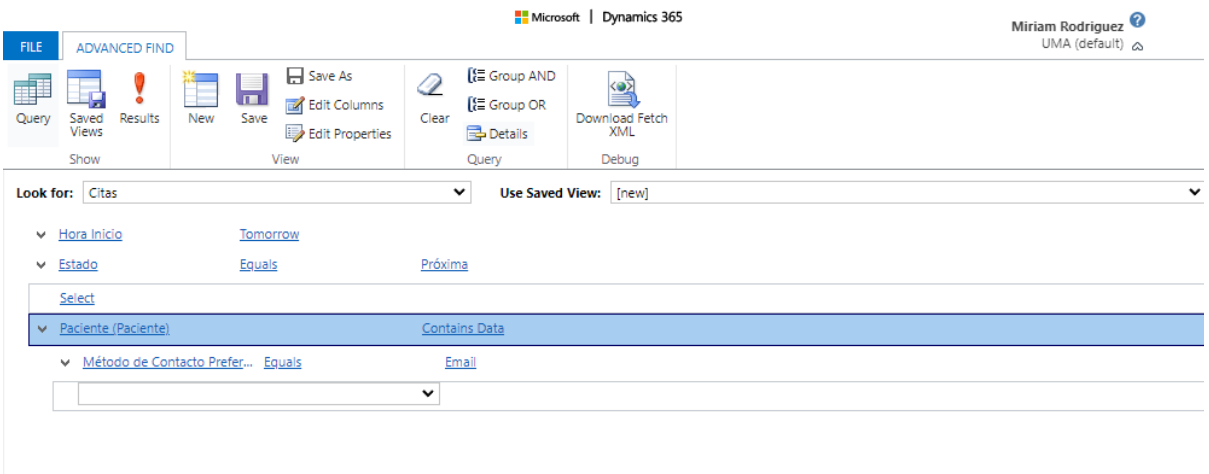

Figura 43. Ventana de búsqueda avanzada

Desde el botón de descarga que vemos en la imagen podemos descargarnos un fichero de formato xml que contenga el filtro que hemos creado. Después podemos pulir ese filtro para facilitar su uso dentro del flujo automatizado, el resultado se muestra en la Figura 44.

```
<?xml version="1.0"?>
- <fetch distinct="false" mapping="logical" output-format="xml-platform" version="1.0">
  - <entity name="appointment">
      <attribute name="subject"/>
      <attribute name="regardingobjectid"/>
      <attribute name="activityid"/>
      <attribute name="instancetypecode"/>
      <attribute name="mra_tipodecita"/>
      <attribute name="scheduledstart"/>
      <order descending="false" attribute="regardingobjectid"/>
    - <filter type="and">
        <condition attribute="scheduledstart" operator="tomorrow"/>
        <condition attribute="statecode" operator="eq" value="0"/>
      </filter>
    - <link-entity name="contact" alias="paciente" link-type="outer" visible="false" to="regardingobjectid" from="contactid">
        <attribute name="mobilephone"/>
        <attribute name="fullname"/>
        <attribute name="firstname"/>
        <attribute name="preferredcontactmethodcode"/>
        <attribute name="emailaddress1"/>
      - <filter type="and">
          <condition attribute="preferredcontactmethodcode" operator="eq" value="2"/>
        </filter>
      </link-entity>
    </entity>
</fetch>
```

Figura 44. Filtro con FetchXML

La siguiente parte que tenemos que tener en cuenta una vez tenemos los datos que necesitamos es conectar con el proveedor de correo de nuestra elección. En nuestro caso hemos creado una conexión con Outlook desde un usuario de nuestro sistema. Una vez el flujo se ejecuta podemos ver en los correos en la bandeja de “enviados” de este usuario. Podemos ver un ejemplo del formato final del correo en la Figura 45.

Figura 45. Correo recordatorio de cita

5. Conclusiones y líneas futuras

En este Trabajo de Fin de Grado hemos diseñado y desarrollado una solución de gestión de clínicas utilizando la plataforma “Low-code/no-code” de Microsoft Power Platform. Hemos demostrado como Power Platform, gracias a su integración con el ecosistema de Microsoft, puede proporcionar una base sólida para la creación de aplicaciones personalizadas.

El uso de Dataverse como fuente de datos, Power Apps para la creación de las distintas aplicaciones y los elementos que las componen como vistas, formularios o lógica como reglas de negocio y Power Automate para la automatización de procesos, han facilitado el desarrollo de una solución completa y robusta.

Uno de los mayores beneficios es la madurez de la plataforma, que presenta numerosas opciones que mejoran la calidad del desarrollo, como los editores personalizados y prefijos, que simplifican enormemente la administración y organización de los componentes personalizados; la gestión de versiones integrada en el sistema, que nos libera de tener que usar gestores de versiones externos; y una estructura de componentes ya definida, facilitando la organización y visualización de los distintos elementos. Además, gracias a la documentación, el foro de dudas y la comunidad que tiene la plataforma en activo se han conseguido solucionar todas las dificultades que se han encontrado durante el desarrollo.

Para la especificación y el desarrollo de los objetivos del proyecto hemos aplicado conocimientos que se han adquirido en asignaturas como Bases de Datos, para construir la estructura inicial de los datos y las relaciones entre estos; Introducción a la Ingeniería del Software, donde aprendemos el uso y las ventajas de las distintas

metodologías de desarrollo; Ingeniería de Requisitos, que enseña a identificar y separar los distintos requisitos además de conocer distintas estructuras que podemos utilizar para presentarlos y describirlos; e Interfaces de Usuario, aplicando las buenas prácticas para asegurar la usabilidad de las interfaces.

En cuanto a trabajos futuros, en este proyecto se ha explorado principalmente la creación de una aplicación basada en modelos (model-driven app), si bien también hemos hecho uso de una aplicación de lienzo (canvas app), como componente adicional a la solución presentada. Las aplicaciones de lienzo son una herramienta muy potente, con mucha profundidad y con una gran capacidad para la personalización, centrándose más en el apartado visual. Sería interesante profundizar en todo lo que ofrecen las canvas app como solución, sobre todo centrándose en el apartado visual, ya que la personalización de las model-driven apps es muy limitada.

Otro aspecto que no se ha tratado con respecto a estas aplicaciones es la personalización de las interfaces. Aunque en principio puedan parecer limitadas, existen herramientas que, por ejemplo, nos permiten ocultar o mostrar nuevos botones en las model-driven apps, que no se han utilizado en el desarrollo de este proyecto.

Con respecto a los aspectos en los que se podría profundizar más con respecto a las model-driven app, destacan el apartado de seguridad, acceso y roles de usuario, ya que la aplicación está pensada para el uso de un usuario, con la posibilidad de ser escalable. Existen múltiples herramientas para gestionar los permisos de los usuarios, lo cual es útil cuando hay muchos interactuando a la vez con la aplicación, y construir un sistema que permita asignar los permisos de forma adecuada a cada uno. Para ello se podrían usar elementos como los roles, los grupos de seguridad, los equipos o las unidades de negocio. Microsoft dispone de documentación sobre cómo utilizar todos estos componentes y cómo construir estructuras para controlar el acceso a los distintos elementos del sistema.

En conclusión, Power Platform es una gran herramienta con la que construir soluciones software, sencilla de utilizar, accesible, personalizable y escalable, con la que hemos podido aplicar una amplia cantidad de los conocimientos aprendidos en el Grado de Ingeniería del Software de manera satisfactoria.

Referencias

1 - Understanding Low-Code No-Code (LCNC) Platforms. (n.d.). National Informatics Centre. Retrieved May 7, 2023, from https://www.nic.in/blogs/understanding-low-code-no-code-lcnc-platforms/

2 - (n.d.). OutSystems: High-Performance Low-Code for App Development. Retrieved May 7, 2023, from https://www.outsystems.com/

3 - (n.d.). The best way to build web apps without code | Bubble. Retrieved May 7, 2023, from https://bubble.io/

4 - Forums - Power Platform Community. (n.d.). Power Platform Community. Retrieved May 7, 2023, from https://powerusers.microsoft.com/t5/Forums/ct-p/PA_Comm_Forums

5 - Información general sobre Microsoft Power Fx - Power Platform. (2023, March 26). Microsoft Learn. Retrieved May 7, 2023, from https://learn.microsoft.com/es-es/power-platform/power-fx/overview

6 - Introducción a las licencias para Microsoft Power Platform - Power Platform. (2023, May 1). Microsoft Learn. Retrieved May 18, 2023, from https://learn.microsoft.com/es-es/power-platform/admin/pricing-billing-skus

7 - ¿Qué es Azure Active Directory? - Microsoft Entra. (2023, March 20). Microsoft Learn. Retrieved May 18, 2023, from https://learn.microsoft.com/es-es/azure/active-directory/fundamentals/active-directory-whatis

8 - Información general de los entornos - Power Platform. (2023, March 26). Microsoft Learn. Retrieved May 18, 2023, from https://learn.microsoft.com/es-es/power-platform/admin/environments-overview

9 - Soluciones en Power Apps - Power Apps. (n.d.). Microsoft Learn. Retrieved May 18, 2023, from https://learn.microsoft.com/es-es/power-apps/maker/data-platform/solutions-overview

10 - Privacidad de datos y cumplimiento - Power Platform. (2023, March 26). Microsoft Learn. Retrieved May 15, 2023, from https://learn.microsoft.com/es-es/power-platform/admin/wp-compliance-data-privacy

11 - (n.d.). Software Médico | Programa Gestión de Clínicas en la Nube. Retrieved May 18, 2023, from https://clinic-cloud.com

12 - (n.d.). ClinicApp.es - Software de gestión de clínicas. Retrieved May 18, 2023, from https://clinicapp.es/

13 - (n.d.). SmartClinic Web – El mejor software médico en la nube. Retrieved May 18, 2023, from https://smartclinicweb.es

14 - Tables in Dataverse - Power Apps. (2022, June 21). Microsoft Learn. Retrieved May 15, 2023, from https://learn.microsoft.com/en-us/power-apps/maker/data-platform/entity-overvie

15 - Naming Conventions in Dynamics NAV - Dynamics NAV. (2018, December 5). Microsoft Learn. Retrieved May 19, 2023, from https://learn.microsoft.com/en-us/dynamics-nav/naming-conventions

16 - About trial environments: standard and subscription-based - Power Platform. (2022, June 21). Microsoft Learn. Retrieved May 18, 2023, from https://learn.microsoft.com/en-us/power-platform/admin/trial-environments

17 - ″. (n.d.). ″ - Wiktionary. Retrieved May 19, 2023, from https://learn.microsoft.com/es-es/dynamics365/customerengagement/on-premises/customize/change-solution-publisher-prefix?view=op-9-1

18 - Rehkopf, M. (n.d.). Historias de usuario | Ejemplos y plantilla. Atlassian. Retrieved March 7, 2023, from https://www.atlassian.com/es/agile/project-management/user-stories

19 - tres c – Con un dibujito se entiende mejor. (2018, January 29). Con un dibujito se entiende mejor. Retrieved March 13, 2023, from https://julibetancur.blog/tag/tres-c/

20 - Ángel, M., & Menzinsky, A. (2015, October 18). ¿Cómo escribir los criterios de aceptación? Blog de un apóstol de Scrum y Kanban. Retrieved March 13, 2023, from https://scrum.menzinsky.com/2015/10/como-escribir-los-criterios-de.html

21 - Radigan, D. (n.d.). El backlog del producto: la lista de tareas pendientes definitiva. Atlassian. Retrieved March 13, 2023, from https://www.atlassian.com/es/agile/scrum/backlogs

22 - Rehkopf, M. (n.d.). Todo lo que necesitas saber sobre los sprints de scrum. Atlassian. Retrieved March 13, 2023, from https://www.atlassian.com/es/agile/scrum/sprints

23 - Use FetchXML to query data (Microsoft Dataverse) - Power Apps. (2022, September 28). Microsoft Learn. Retrieved May 18, 2023, from https://learn.microsoft.com/en-us/power-apps/developer/data-platform/use-fetchxml-construct-query

24 - Client API Reference for model-driven apps - Power Apps. (2022, November 29). Microsoft Learn. Retrieved May 12, 2023, from https://learn.microsoft.com/en-us/power-apps/developer/model-driven-apps/clientapi/reference

25 - Verma, A. (2019, October 7). Calling Namespace Javascript function in Dynamics CRM. Learning Refresh. Retrieved May 16, 2023, from https://learningrefresh.home.blog/2019/10/07/calling-namespaced-javascript-function-in-dynami

Apéndice A

Manual de Instalación

Requerimientos

Lo primero que tenemos que hacer es exportar nuestra solución desde el entorno de desarrollo. Para ello, desde la pantalla principal de soluciones seleccionaremos la solución “Gestion Clinic App” y pulsaremos el botón de “exportar”. Al hacerlo se nos abrirá un menú lateral donde nos recomienda que hagamos algunas acciones, como vemos en la Figura A1. Ejecutaremos la acción de “Publicar todos con cambios” pulsando el botón “publicar”, de esta manera nos aseguramos que la aplicación contenga los últimos cambios.

Figura A1. Menú lateral exportación - Antes de exportar

Una vez publicados los cambios, continuaremos con el proceso pulsando el botón “Siguiente”. La siguiente pantalla del menú, Figura A2, nos dejan ver y editar los detalles de la versión de nuestra solución y además nos dan a elegir entre dos opciones a la hora de importarla, junto con una breve explicación del uso de cada una. Elegiremos la que más nos convenga según si vamos a mover la solución a un nuevo entorno de desarrollo o a un entorno de producción.

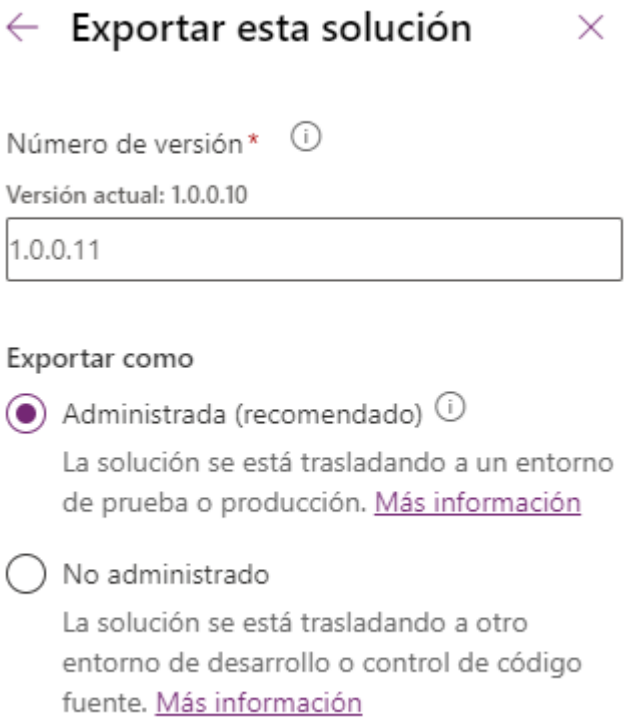

Figura A2. Exportación de la solución como administrada o no administrada

Al cerrar el menú lateral, volveremos a ver la pantalla principal al completo y observaremos que ahora se ve una notificación como la Figura A3, que nos avisa de que la solución se está exportando.

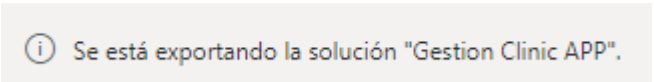

Figura A3. Notificación de exportación

Cuando termine de cargar, podremos pulsar el botón “descargar” desde la propia notificación, que ahora se ve coloreada en verde en la Figura A4.

Figura A4. Notificación de descarga de la solución

Se activará automáticamente la descargar de un archivo .zip. Este archivo es el que seleccionaremos para importar los componentes de la solución a nuestro nuevo entorno. Para ello de nuevo, y dentro del otro entorno, navegaremos a la pantalla principal de soluciones y, sin seleccionar ninguna, pulsaremos en el botón “Importar solución”. Se abrirá otro menú lateral, Figura A5, donde podremos pulsar el botón “examinar” para seleccionar el archivo que acabamos de descargar.

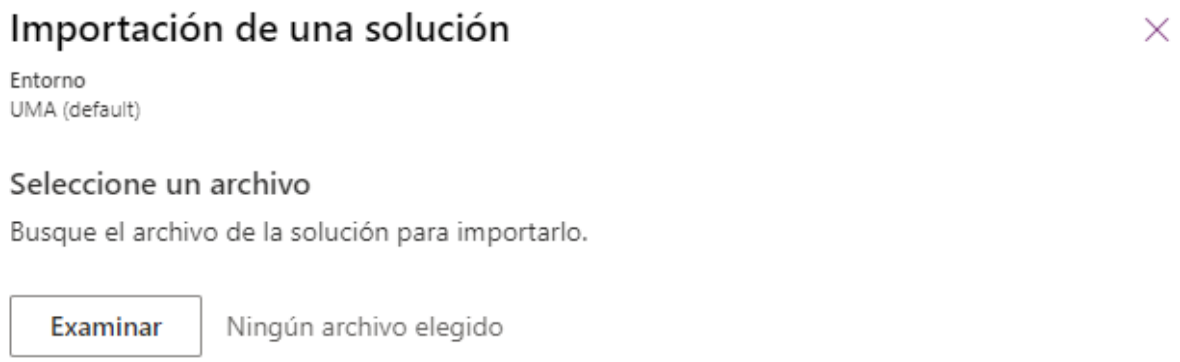

Figura A5. Importar solución

Al navegar a la siguiente pantalla del menú lateral, Figura A6, podemos ver los detalles de la solución que estamos importando, y en caso de que haya algún error en la importación, como que haya algunos componentes con dependencias y esas dependencias no están dentro de la solución, nos mostrará un mensaje de error y nos avisará de las dependencias que tenemos que solucionar para finalizar la importación. Solo habría que introducir los componentes que se mencionan en la solución y volver a repetir los pasos para su exportación e importación.

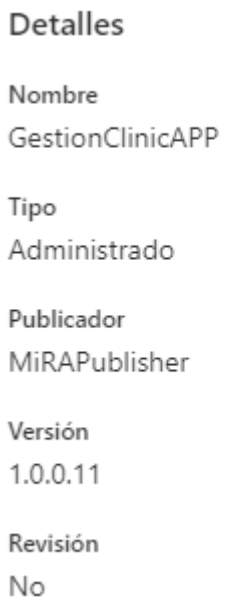

Figura A6. Detalles de la solución

Si hemos realizado bien todos los pasos finalmente podremos ver que nuestra solución se ha importado en el nuevo entorno de forma satisfactoria. Ya solo tendrían que crearse o configurar los usuarios y sus permisos desde el admin center, si no se ha hecho ya junto con la creación del entorno, para que dispongan de los credenciales y permisos necesarios para acceder a la aplicación.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga