UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Seguimiento de requerimientos para gestionar, medir, evaluar y mejorar

Presentado por Pablo Ahíta del Barrio
en Universidad de Burgos — 6 de julio de 2023
Tutor: Pedro Luis Sánchez Ortega

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Pedro Luis Sánchez Ortega, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Expone:

Que el alumno D. Pablo Ahíta del Barrio, con DNI 71566290L, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado **Seguimiento de requerimientos para gestionar,medir, evaluar y mejorar.**

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 6 de julio de 2023

Vº. Bº. del Tutor:

SANCHEZ ORTEGA, PEDRO LUIS (FIRMA)

Firmado digitalmente por SANCHEZ ORTEGA, PEDRO LUIS (FIRMA)
Fecha: 2023.07.06 10:52:30 +02'00'

D. Pedro Luis Sánchez Ortega

Resumen

En este proyecto de final de grado se ha desarrollado una aplicación Android con el objetivo de sustituir a un antiguo script de nombre *OTEA*, que almacena localmente en cada ordenador la información sobre los test de indicadores o características de calidad de vida realizados por la *Fundación Miradas*.
La forma en la que se trasladaba el programa entre diferentes computadores era mediante un CD-ROM que tenía que llevar a mano siempre la persona encargada de hacer el test de indicadores, tecnología que se ha sustituido por una arquitectura cliente-servidor, por lo que la funcionalidad de esta aplicación Access se traslade a una aplicación para Android que se conecta a una aplicación web desplegada en Azure que a su vez se comunica con la base de datos para realizar todas las consultas que se le solicitan.
En cuanto a la arquitectura, el cliente es la propia aplicación de Android desarrollada en *Java*, con su interfaz gráfica y su código interno para manejar las peticiones HTTP al servidor mediante *Retrofit* y *OkHttp* con la ayuda de *ASyncTask* que ejecuta esas órdenes en un hilo diferente al de la propia aplicación y para realizar toda la funcionalidad esperada dependiendo del tipo de usuario que maneje dicha aplicación. Mientras tanto el servidor se ha desarrollado en *C#*, utilizando para el manejo de las peticiones HTTP la API de *ASP.NET*, la cual se encarga de recibir las peticiones por parte del cliente y de enviar la respuesta del endpoint solicitado, utilizando el paquete *System.Data.SqlClient* para comunicarse con la base de datos y el paquete *Newtosnsoft.Json* para transformar a JSON el objeto y serializar sus parámetros para que el cliente mediante *Retrofit* los deserialice y los utilice para crear las instancias correspondientes.
Los tipos de usuarios distinguidos por la aplicación ayudan a distinguir correctamente las acciones que cada uno de ellos puede realizar en la aplicación. En primer lugar tenemos los **administradores** que controlan las tablas de la base de datos, los **usuarios de organización evaluada** que reciben los test de indicadores para luego mostrar su evolución y los **usuarios de la *Fundación Miradas***, que se encargan de realizar diferentes evaluaciones de indicadores a cualquier organización evaluada y de añadir las nuevas organizaciones evaluadas.

Descriptores

Android, Azure, Azure SQL, C#, cliente-servidor, evidencias, indicadores, JSON, SQL-Server, servicio web.

Abstract

In this final degree project an Android app has been developed with the goal of subsituting an old Access Script named *OTEA*, that stores locally in every computer all the information about indicators or life quality features realized by *Fundación Miradas*.
The way used to move the softwares between computers was using a CD-ROM that the person that realizes the indicator test had to take that with him, technology substituted by an architecture client-server, so the old Access application funcionality was finally to an Android app that comunicates with an web-app deployed in Azure that communicates with a Azure SQL database to realize all the queries ordered by the client.
Regarding the architecture, the client is the proper Android app developed in *Java*, including its graphic interface and the internal code that controls the HTTP requests using *Retrofit* and *OkHttp*, both being helped by *ASyncTask* that runs all the calls to the server in an app-separated thread, becaming in a huge help to realize all the funcionality depending of the user's type. Meanwhile the server was developed using *C#*, using *ASP.NET* to control all the HTTP requests that are received from the client and then sending the requested endpoint's response, using for that the *System.Data.SQLClient* package to communicate with the database and the *Newtonsoft.Json* to convert to Json the object and serialice its parameters before the client deserialices it using *Retrofit* and then uses it to build the corresponding instances.
The types of users distinguished by the application help to distinguish correctly the actions that each one of them can carry out in the application. First of all we have the **administrators** who control the database tables, the **evaluated organization users** that receive the indicator tests to later show their evolution and the ***Fundación Miradas*'s users**, who are in charge of carry out different evaluations of indicators to any organization evaluated and to add the new organizations evaluated.

Keywords

Android, Azure, Azure SQL, C#, client-server, evidences, indicators, JSON, SQL-Server, web-service.

Agradecimientos

En primer lugar agradezco a mi familia, en especial a mis padres **Diego y Yolanda** y a mis **abuelos Lola y Salva**, por ser el principal motor de mi vida en todo momento gracias a su constante apoyo y cariño, en el que también están mis mejores amigos de toda la vida **Antonio y Adrián**, con quienes he aprendido el significado de la verdadera amistad.
Asimismo quiero expresar mi gratitud a los docentes que me han apoyado a lo largo de mi etapa universitaria, a mi tutor **Pedro Luis Sánchez** por su constante guía y apoyo en este proyecto y al profesor **Raúl Marticorena** por aportarme desde su experiencia los conceptos necesarios para un mejor desarrollo del mismo.
También agradezco a la *Fundación Miradas* su plena confianza en mí y su involucración en el proyecto, a su director **Miguel Gómez**, a su analista **Fernando Terradillos** y **Jose Luis Cuesta**, director de la *Cátedra Miradas por el Autismo*.
A su vez quiero agradecer a **Sonia Rodríguez** su apoyo en todo momento y por acompañarme y guiarme a lo largo de mi vida académica y personal.
Asimismo muestro mi agradecimiento a los docentes que me han apoyado y me han animado a lo largo de todas las etapas educativas que he ido superando hasta la fecha.
También muestro mi agradecimiento a **Natividad de Juan** su constante orientación, respaldo y asesoramiento durante toda mi etapa universitaria.
A su vez muestro mi agradecimiento a **Cristina Barriuso** por su fe en mí y a **Jennifer Terceño** por su constante apoyo durante todos estos años.
Por último, quiero recordar y agradecer al profesor **Joaquín Seco** por haberme aportado sus conocimientos y experiencia para este proyecto.

Índice general

Índice de figuras

Índice de tablas

Introducción

En este proyecto final de grado se ha desarrollado una aplicación compatible con Android denominada *OTEA (**O**rganizaciones que prestan apoyo a personas con **T**rastorno del **E**spectro **A**utista)* para la fundación especializada en trastorno del espectro autista con sede en la ciudad de Burgos denominada *Fundación Miradas*, la cual a fecha de este año 2023 celebra el décimo aniversario de su creación por parte de *Autismo Burgos*.
El proyecto como tal consiste en el paso de una aplicación en un fichero de Microsoft Access grabado en un CD-ROM, donde se guardaban todos los datos y registros, y se pasaba de mano en mano a cada uno de los ordenadores que lo necesitan. El cambio a una implementación más moderna de esta aplicación se antoja necesario, ya que en primer lugar el uso de los discos ópticos ya no es habitual en la informática de la actualidad, puesto que cada vez más equipos carecen de una bandeja compatible con estos discos, a parte de que la versión utilizada data del año 2009 sobre una versión del paquete de *Microsoft Office 2007*, quedando dichas tecnologías como totalmente obsoletas.
Una solución a este problema nos la proporciona la computación en la nube, el uso masificado de los dispositivos móviles que permite la existencia de las denominadas aplicaciones multidispositivo y el auge de las implementaciones cliente-servidor, aspectos que quedan perfectamente cubiertos gracias a la implementación de una aplicación web y de una base de datos en *Microsoft Azure*, permitiendo la comunicación entre las diferentes tablas de la base de datos y la web app que recibe las peticiones correspondientes a cada endpoint que realiza para posteriormente devolver la consulta de la base de datos correspondiente y enviarla a posteriori al cliente. En cuanto a la aplicación, se ha decidido que se va a implementar en primer lugar para Android, debido a que es la plataforma de dispositivos móviles más utilizada en el mercado a nivel mundial por el bajo costo de los dispositivos que cuentan

con el mencionado sistema operativo en comparación con su competencia, cubriendo una cuota de mercado importante.

Objetivos del proyecto

En cuanto a los objetivos a cumplir en este proyecto, se tienen que tener en cuenta los cuatro principios que debe tener todo software, siendo éstos el *control*, la *comodidad*, la *eficiencia* y la *evolución*, sirviendo de base para poder mencionar los objetivos que se cumplen y se tienen que seguir cumpliendo en la aplicación de *OTEA*, para garantizar el mejor funcionamiento posible de la misma dentro de cada uno de esos cuatro principios. Por lo tanto, los objetivos a esperar dentro del software son los siguientes:

- **Objetivos relacionados con el principio del control:** La aplicación *OTEA* debe garantizar que los usuarios tengan un control total sobre las acciones que realizan dentro de la aplicación y los resultados que tengan de las mismas. En este caso se busca que los usuarios de la Fundación Miradas puedan realizar los test de indicadores marcando las evidencias que se cumplen, para luego hacer que los usuarios de las organizaciones evaluadas puedan observar los resultados de la puntuación total de cada test y sus respectivos gráficos. Además los usuarios deben tener el poder de personalizar la propia aplicación en la medida de lo posible, como elegir el idioma de la misma, algo que se consigue gracias a que se ha realizado la internacionalización a tres idiomas (español, inglés y francés).
- **Objetivos relacionados con el principio de la comodidad:** La aplicación *OTEA* debe ser una aplicación intuitiva y fácil de aprender a manejar, pudiendo ser utilizado por usuarios de todos los niveles, desde los usuarios casuales hasta los usuarios expertos. En caso de que se necesite ayuda, se tiene que ofrecer un manual de instrucciones de la misma, priorizando la existencia de vídeos junto con un manual escrito.

La aplicación también tiene que estar preparada para que tenga el soporte para los tres tipos de usuarios: **administradores**, **usuarios de organizaciones evaluadas** y **usuarios de la *Fundación Miradas***, teniendo cada uno de los usuarios sus funcionalidades muy bien marcadas desde el principio.

- **Objetivos relacionados con la eficiencia:** La aplicación *OTEA* debe garantizar unos tiempos de respuesta en las peticiones a la base de datos en formato HTTP lo más rápidas y eficientes posible, haciendo que no se consuman una cantidad enorme de recursos en el dispositivo, en cuanto a espacio del disco duro y uso del procesador y de la memoria RAM. Gracias al correcto manejo de la aplicación en aspectos de hardware, se espera que sea de comportamiento fluido y con los tiempos de espera a las respuestas de las peticiones HTTP lo más cortos posible.

- **Objetivos relacionados con la evolución:** La aplicación *OTEA* debe ser una aplicación que sea siempre susceptible a diferentes cambios y mejoras en la funcionalidad de la misma, adaptándose siempre al avance constante de la computación en la nube y de las tecnologías móviles. Al desarrollador no le tiene que temblar el pulso para poder tomar decisiones arriesgadas que puedan desembocar en la adición de dichas mejoras y características adicionales, cumpliendo así con el carácter ambicioso que tiene la *Fundación Miradas* para realizar su cometido.

Conceptos teóricos

En dicho apartado se va a desarrollar en que consiste el proyecto anteriormente mencionado denominado *OTEA*, el cual se ha obtenido de la guía escrita por el profesor de la Universidad de Burgos *Jose Luis Cuesta Gómez.* Dicho capítulo sirve como punto de referencia para poder mostrar el trabajo que se ha ido añadiendo poco a poco en la aplicación, y la funcionalidad que se espera de la misma. Cabe resaltar que en comparación con el momento en el que se publicó dicha guía, los indicadores han sido modificados y su número ha incrementado en el momento actual, aunque tanto antes como ahora la finalidad de los indicadores sea la misma.

3.1. Guía de Indicadores de Calidad de Vida

La Guía de Indicadores de Calidad de Vida es un instrumento de evaluación desde una perspectiva objetiva (basada en las condiciones de vida de los afectados). Contempla aquellos factores contextuales referidos a las organizaciones donde se integran las personas con TEA (*Trastorno del Espectro Autista*), que pueden incidir significativamente, de forma directa o indirecta, en su calidad de vida. Este instrumento consta de *68 indicadores* agrupados en *seis ámbitos* que definen los diferentes aspectos que puede tener una organización y que debemos contemplar para evaluar su impacto en la calidad de vida de las personas:

- **Calidad referida a la persona:** En este ámbito no sólo se valoran aspectos organizativos que tienen un impacto directo sobre la calidad de vida referida de la persona con TEA, atendiendo a cada una de las dimensiones propuestas por Robert Schalock (1996) (*bienestar físico*, *bienestar emocional*, *bienestar material*, *relaciones interpersonales*,

desarrollo personal, *derechos*, *autodeterminación* e *inclusión social*), sino en la de aquellas personas que conviven con ellas y que conforman sus contextos vitales más cercanos: familia y profesionales. Es evidente que unas condiciones de vida saludables en familias y profesionales genera directamente, entre otros aspectos positivos, una mejor relación con la persona con TEA. Cuando una organización facilita la mejora de la calidad de vida de familias y profesionales reforzando vías de motivación, implicación y reconocimiento, está generando un impacto similar en las personas con TEA.

- **Identificación de las necesidades y preferencias / Elaboración y seguimiento de los planes de desarrollo personal:** Un proceso de detección de necesidades, planificación de metas y diseño de apoyos, debe realizarse de forma coordinada, implicando a todas aquellas personas significativas en la vida de la persona con TEA, y facilitar el que esta tenga un papel realmente activo, de forma que su plan de desarrollo responda a sus intereses, capacidades y preferencias.

- **Formación de profesionales:** A un determinado perfil personal el profesional debe sumar un amplio conocimiento del autismo y de cada persona con la que interviene, además del dominio de diferentes técnicas y metodologías que faciliten su intervención, la coordinación de apoyos en diferentes contextos y la adaptación a las necesidades e intereses de la persona.

- **Estructura y organización:** En este ámbito se valoran aspectos referidos a los agrupamientos, organización del trabajo, horarios, comunicación/coordinación y análisis de situaciones susceptibles de mejora que pueden facilitar el bienestar de las personas con TEA.

- **Recursos y servicios:** La respuesta a las necesidades de las personas con TEA requiere de una determinada provisión de recursos personales y materiales, y de su óptima organización.

- **Relación con la comunidad / Proyección social:** La inclusión social es una de las dimensiones claves de la calidad de vida, y en este ámbito se valoran diferentes aspectos referidos a cómo la organización se proyecta hacia el exterior y facilita la participación en la comunidad de las personas con TEA.

La utilización de indicadores como medida de evaluación es útil para mejorar resultados, puesto que su medida es significativa e interpretable,

y permiten la recogida de datos sin excesivo esfuerzo y, como en el caso de esta Guía, están basados en una teoría y se deciden por consenso. En esta línea se configura la Guía de Indicadores de Calidad de Vida, como un instrumento que pretende ser sensible a los apoyos y condiciones de las organizaciones en relación a la persona y a su inclusión en la comunidad, necesarios para mejorar la calidad de vida. Cada indicador consta de cuatro evidencias, es decir, cuatro realidades fácilmente observables que nos van a ayudar a hacer cuantificable el indicador, y a poder comprobar si este se cumple o no con un mismo criterio de valoración objetivo para todos los evaluadores.

3.2. Metodología de aplicación

Para la aplicación de la Guía de Indicadores de Calidad de Vida se deberá determinar:

- **Equipo Consultor:**Un *Equipo Consultor del Plan de Calidad de Vida (ECPCV)* es el conjunto de personas encargado de realizar la valoración de las variables, traducidas en los indicadores y evidencias que conforman la Guía, que desde la organización pueden intervenir en la calidad de vida de las personas con TEA. El equipo consultor estará compuesto, al menos, por:

  - ***Evaluador principal*** Profesional externo a la Organización donde se va a realizar la evaluación y con formación y experiencia en la aplicación de instrumentos relacionados con sistemas de gestión de calidad. Este profesional o evaluador principal será el encargado de dirigir el proceso de aplicación de los indicadores y contrastar cada uno de ellos definiendo también el papel que para esta tarea pueden desempeñar los demás componentes del equipo consultor.
  - ***Responsable de la organización o del servicio donde se aplicará la guía:*** Su función principal será servir de guía e intermediario entre el evaluador principal y la Organización o el servicio, facilitando a éste el acceso a la información y a toda persona que pueda facilitar evidencias que permitan contrastar los indicadores. Esto resulta especialmente necesario en el caso de grandes organizaciones donde la persona responsable no conozca en profundidad todos los servicios. El responsable del servicio, además de aportar la información propia del cargo, ejercerá las

funciones de "secretario", tomando nota de los acuerdos a los que se llegue.

- ***Familiar de una de las personas con TEA:*** Este familiar debe conocer la Organización y ser designado por ésta para representarla. Su papel se centrará en facilitar la evaluación, ayudando a encontrar evidencias que permitan comprobar cada uno de los indicadores.
- ***Un profesional de atención directa:*** Designado por la Organización, su función consistirá en aportar y facilitar el acceso a la información, guiando la búsqueda de evidencias que ayuden a evaluar cada uno de los indicadores. Además del conocimiento de la Organización o servicio, sería muy oportuno que este profesional tuviera conocimientos o experiencia en el ámbito de los sistemas de gestión de calidad. Igualmente facilitará, siempre que sea posible, que las propias personas con TEA del servicio o la Organización, aporten datos que puedan asegurar la exploración de evidencias.

- **Planificación de la evaluación:** Esta fase se iniciará con una visita previa a la Organización, en la que todo el equipo consultor tendrá la oportunidad de conocerse y planificar el proceso. El Evaluador Principal, acompañado del resto del equipo, conocerá así la Organización, lo que le permitirá situarse para desarrollar la evaluación. En esta visita previa conviene que se defina el calendario de las posteriores sesiones de trabajo y se realice una estimación del tiempo necesario para la aplicación del instrumento. Aunque debemos entender que este periodo es estimativo, debe servir de referencia para evitar el retraso excesivo que pueda suponer un cambio de las condiciones evaluadas y suponga una pérdida del trabajo desarrollado. La experiencia desarrollada nos orienta a que la valoración se realice en las siguientes condiciones:

  - Al menos tres reuniones, de dos horas cada una.
  - En un periodo no superior a un mes.

  Siempre que sea posible, el primer día de reunión se fijarán las fechas en las que se reunirá el equipo para realizar la evaluación.

- **Sesiones de trabajo:** Las reuniones del equipo consultor tendrán lugar en el centro en el que las personas con TEA desarrollan prioritariamente la actividad o desde el que se lleva a cabo la planificación de apoyos y el seguimiento. Una vez en la organización, el equipo consultor tendrá la posibilidad de solicitar la presencia puntual de

otros profesionales de referencia en los diferentes ámbitos que engloba el instrumento: formación, planificación, organización y recursos..., para solicitar información que les ayude a comprobar cada uno de los indicadores a través de las evidencias. Para encontrar evidencias podemos recurrir a distintas vías:

- ***Observación directa***.
- ***Análisis de documentación***.
- ***Contacto e intercambio con los profesionales***.
- ***Si fuera posible, y el equipo lo estima oportuno, consulta a personascon TEA***.

La guía de indicadores y la plantilla de registro de los mismos será sustituida por el registro en la base de datos, la cual facilita mucho el proceso mencionado con anterioridad.

La valoración de cada una de las evidencias será *0 o false* si no se ha alcanzado y *1 o true* en caso contrario.

Las decisiones tomadas por el Equipo Consultor deben responder al mayor grado posible de acuerdo. En cualquier caso, ante desacuerdos en la valoración de alguna de las evidencias, se deberá contar con:

- ***El acuerdo de 3 de los 4 miembros del equipo consultor.***
- ***Siempre deberá contar con el acuerdo del evaluador principal.***

3.3. Tabulación de datos

La información que se ha recogido dará lugar a dos tipos de información:

- Por una parte, la organización dispondrá de un Gráfico del Servicio u Organización que permitirá conocer sus debilidades y fortalezas en relación a la calidad de vida. El gráfico representa todos los indicadores que conforman la guía, distribuidos en diferentes franjas de acuerdo a su grado de importancia. Los indicadores se distribuyen en cuatro niveles de importancia o interés:
  - De interés fundamental.
  - De interés alto.

- De interés medio.
- De menor interés.

En el gráfico es posible consultar:

- El grado de consecución de cada uno de los indicadores.
- Importancia de cada indicador en relación al conjunto.
- Cuántos indicadores tiene la Organización conseguidos, en proceso, o no conseguidos.
- Cómo se distribuyen los indicadores según los niveles de interés. La información que aporta nos ayudará a tener una rápida y clara visión de la situación en la que se encuentra la organización facilitándonos la realización del Informe Final y la planificación de las acciones de mejora si fueran necesarias.

Con los datos obtenidos, la persona responsable de la construcción del gráfico, deberá aplicar los siguientes criterios en la valoración de cada indicador:

- En color rojo se marcarán los **indicadores no conseguidos**, que se trata de todo aquel indicador en el que como máximo se cumple una evidencia.
- En color amarillo se marcarán los **indicadores en proceso**, que son aquellos indicadores en el que se cumplen 2 o 3 evidencias.
- En color verde se marcarán los **indicadores conseguidos**, los cuales son aquellos en los que se cumplen todas las evidencias.

Una vez realizado este proceso la Organización contará con un gráfico de resultados de la guía de indicadores de calidad de vida como el ejemplo que recogemos a continuación:

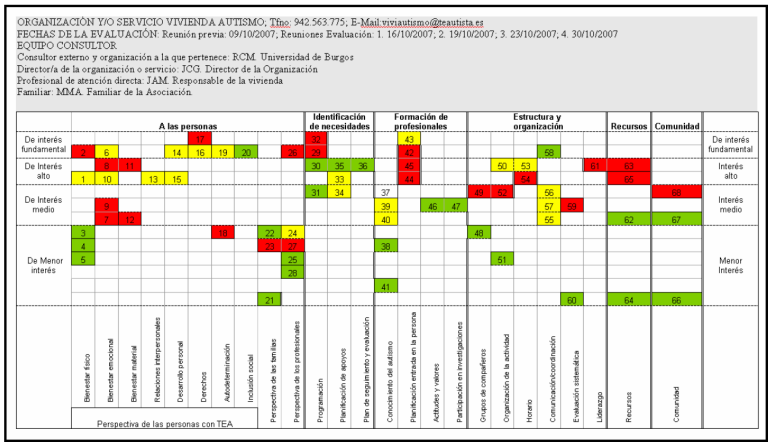

Figura 3.1: Ejemplo de gráfico de indicadores

- Por otra, obtendrá una puntuación global de la organización. Partiendo del gráfico anterior anotaremos una puntuación para cada uno de los indicadores, de acuerdo a su nivel de interés, cuya suma nos permitirá obtener la puntuación global de la organización.

Nivel		Nº de indicadores anotados	Multiplicar por	
De interés fundamental			5	
			4	
			0	0
Interés alto			4	
			3	
			0	0
Interés medio			3	
			2	
			0	0
Menor interés			2	
			1	
			0	0
TOTAL				

Figura 3.2: Modelo de tabla de puntuación total

La puntuación global de la organización nos aporta información sobre el nivel en que ésta se encuentra respecto a la aplicación de la Guía de indicadores. Un ejemplo de la aplicación de dicha tabla es la siguiente:

Nivel		Nº de indicadores anotados	Multiplicar por	
De interés fundamental		2	5	10
		5	4	20
		4	0	0
Interés alto		3	4	12
		7	3	21
		8	0	0
Interés medio		5	3	15
		6	2	12
		7	0	0
Menor interés		14	2	28
		1	1	1
		3	0	0
TOTAL				**119**

Figura 3.3: Ejemplo de tabla de puntuación total

Esta conclusión del proceso la deberá realizar el Responsable de la Organización, para posteriormente convocar de nuevo al Equipo Consultor, encargado este de elaborar el Informe Final que servirá de referencia para la elaboración posterior del Plan de Mejora por parte de los responsables de la organización o servicio. La puntuación global obtenida nos permite comprobar el nivel en el que se encuentra la organización:

Puntuación	Nivel	Significado
198 – 246	Excelente	La organización promueve un alto nivel de calidad de vida para las personas con TEA en todos los ámbitos, y asume la necesidad de desarrollar un proceso de mejora continua.
149 – 197	Muy bueno	La organización promueve calidad de vida para las personas con TEA en todos sus ámbitos, aunque se observan algunas cuestiones organizativas que deberían corregirse. Se plantean algunas observaciones cuyo cumplimiento ayudarán a elevar el nivel.
100 – 148	Bueno	La organización promueve calidad de vida para las personas con TEA, aunque se proponen algunas sugerencias de mejora que deben tenerse en cuenta poniendo especial atención en aquellas referidas a los indicadores de mayor interés.
51 – 99	Mejorable	Se requiere una nueva aplicación de la Guía que permita revisar de nuevo el cumplimiento de los indicadores no conseguidos. La Organización puede mejorar la calidad de vida que facilita a las personas con TEA, y para ello debe diseñar un Plan de Mejora y volver a aplicar nuevamente la Guía transcurrido el tiempo necesario para poder implantarlo.
0 – 50	Muy mejorable	Se requiere una nueva aplicación de toda la Guía. La organización o el servicio no tiene implantado el modelo de calidad de vida. Es fundamental un Plan de Mejora, su aplicación inmediata y la posterior reevaluación.

Tabla 3.1: Valores de la suma total de los indicadores

3.4. Informe final

El proceso de aplicación de la Guía de Indicadores concluirá con la elaboración y presentación de un Informe Final por parte del Equipo Consultor, cuyo objetivo es orientar el consiguiente Plan de Mejora cuya elaboración es responsabilidad de la dirección de la Organización o servicio. El Informe Final incluirá observaciones generales, indicadores clave que orienten sobre qué aspectos mínimos deben tenerse en cuenta para mejorar el nivel en los diferentes ámbitos, pautas de mejora, así como una propuesta de fecha de la revisión en los casos en que fuera pertinente debido a que la Organización o el servicio no cumple un número significativo de indicadores en relación al total, en determinados niveles de interés, o en alguno de los ámbitos que engloba la Guía y que el Equipo Consultor considere necesario mejorar.

Técnicas y herramientas

4.1. *Microsoft Azure*

Microsoft Azure es una plataforma que proporciona diferentes servicios en la nube, permitiendo la construcción, prueba, despliegue y administración de los mismos. Esta plataforma fue anunciada en el año 2010 por Microsoft con el nombre de *Windows Azure*, pasando a su denominación actual el 25 de marzo del año 2014.
En este proyecto se utiliza *Azure* para desplegar la aplicación web que es utilizada para la gestión de las operaciones de la base de datos. La aplicación web está desplegada mediante el servicio denominado *Web Service*, el cual es utilizado para el despliegue de aplicaciones web hechas en diferentes tecnologías de diferentes lenguajes de programación, destacando C#, Java y Python. Mientras tanto, la base de datos es implementada utilizando *SQL Server* mediante el servicio denominado *Azure SQL*, el cual proporciona la cadena de conexión necesaria para el servicio web, crea el servidor de base de datos y proporciona el soporte necesario para la ejecución de consultas en SQL.
Para poder utilizar *Microsoft Azure*[8], es preciso contar con una cuenta con la cual se tienen ciertos servicios de forma gratuita, algunos de ellos de forma permanente y otros tantos durante un total de 12 meses. Adicionalmente a esta base, se pueden añadir servicios o mejorar los ya existentes a partir de diferentes niveles de suscripción a los mismos, los cuales se ajustan a las necesidades que tengan los usuarios u organizaciones para sus actividades. Para cubrir dichas actividades, *Microsoft Azure* proporciona un crédito inicial de 200$ para utilizarse durante el primer año, el cual puede ampliarse de forma opcional eligiendo un método de pago, ya sea mediante transferencia bancaria o mediante tarjeta de crédito o de débito.

En primera instancia, el inicio del despliegue de la aplicación en *Microsoft Azure* se ha realizado con la propia cuenta de la Universidad de Burgos [1], el cual permite el uso de *Azure for Education* [6] ya que se trata de uno de los diferentes servicios de *Microsoft 365* del cual disponen los alumnos de manera gratuita, la cual destaca por un crédito inicial de 100$ y por no necesitar introducir un método de pago para la creación de la cuenta. A posteriori, la implementación definitiva de la aplicación se ha realizado en el propio servidor de la *Fundación Miradas*, disponiendo para ello con una cuenta diferente a la de la Universidad de Burgos.
En cuanto al aprendizaje de la herramienta, se dispone de una herramienta de aprendizaje denominada *Microsoft Learn* [9], la cual consta de diferentes cursos autodidactas e interactivos sin restricción alguna en cuanto a horarios, permitendo un aprendizaje adaptado al ritmo que cada usuario tenga y al tiempo que éste le pueda dedicar a los mismos. *Microsoft Learn* también dispone de diferentes herramientas alternativas para incrementar la experiencia y el aprendizaje del usuario, como la presencia de foros y la búsqueda de documentación técnica sobre las diferentes herramientas de *Microsoft*. En última instancia, *Microsoft Learn* también ofrece la obtención de diferentes certificados oficiales de *Microsoft*, adaptados al rol que desempeña cada usuario en el equipo de trabajo, todo ello gracias a los cursos y módulos de aprendizaje correspondientes, aunque para conseguir esa certificación se necesita realizar un examen previo pago de una tasa de inscripción al mismo.

4.2. Entornos de desarrollo utilizados

Microsoft Visual Studio

Para el desarrollo del lado del servidor se ha optado por utilizar *Microsoft Visual Studio Community*, la cual se trata de la versión más básica de este entorno de desarrollo, si no consideramos que se tiene *Microsoft Visual Studio Code*. Dicho entorno de desarrollo es gratuito, por lo que no supone ningún coste adicional con respecto a sus versiones *Enterprise* y *Professional*, los cuales sí que tienen una licencia de pago con posibilidad de probar el software de manera gratuita.
Visual Studio es una herramienta de desarrollo eficaz que permite completar todo el ciclo de desarrollo en un solo lugar. Es un entorno de desarrollo integrado completo que permite la escritura, edición, depuración y compilación del código y, luego, su posterior implementación. Aparte de la edición y depuración del código, Visual Studio incluye compiladores, herramientas de

finalización de código, control de código fuente, extensiones y muchas más características para mejorar cada fase del proceso de desarrollo de software. Visual Studio proporciona a los desarrolladores un entorno de desarrollo enriquecido para desarrollar código de alta calidad de forma eficaz y colaborativa:[5]

- Instalador basado en cargas de trabajo: instale solo lo que necesita.
- Herramientas y características de codificación eficaces: todo lo que necesita para compilar sus aplicaciones en un solo lugar.
- Compatibilidad con varios lenguajes: código en C++, C#, JavaScript, TypeScript, Python, etc.
- Desarrollo multiplataforma: compilación de aplicaciones para cualquier plataforma.
- Integración del control de versiones: colaboración en el código con compañeros de equipo.

El motivo por el cual se ha utilizado este entorno de desarrollo para el lado del servidor es por los cursos de *Azure Learn* que se han ido siguiendo para el aprendizaje de las herramientas de Azure para este tipo de aplicaciones, aunque también pueda utilizarse *Microsoft Visual Studio Code* para la programación del lado del servidor. Al haber realizado el aprendizaje de esta manera, no se necesita realizar aprendizaje de otras herramientas, ayudando a reforzar dicha decisión.

Por lo tanto, las principales características de este entorno de desarrollo son las siguientes:[3]

- **Instalación modular:** En el instalador modular de Visual Studio, se eligen y se instalan exclusivamente las cargas de trabajo que sean necesarias. Las cargas de trabajo son grupos de características que los lenguajes de programación o las plataformas necesitan para funcionar. Esta estrategia modular ayuda a reducir la superficie de instalación de Visual Studio, por lo que se instala y actualiza más rápido.
- **Creación de aplicaciones de Azure habilitadas para la nube:** Visual Studio ofrece un conjunto de herramientas para crear fácilmente aplicaciones habilitadas para la nube de Microsoft Azure, permitiendo

la configuración, compilación, depuración, empaquetado e implementación de aplicaciones y servicios de Azure directamente desde el entorno de desarrollo integrado (IDE). Para obtener las plantillas de proyecto y las herramientas de Azure, se tiene que seleccionar la carga de trabajo Desarrollo de Azure al instalar Visual Studio.

- **Creación de aplicaciones web:** Visual Studio puede crear aplicaciones web mediante ASP.NET, Node.js, Python, JavaScript y TypeScript. Visual Studio admite muchos marcos web, como Angular, jQuery y Express. ASP.NET Core y .NET Core funcionan en los sistemas operativos Windows, Mac y Linux. ASP.NET Core es una actualización principal a MVC, WebAPI y SignalR. ASP.NET Core se diseñó desde la base para ofrecer una pila de .NET eficiente y componible, con el fin de compilar servicios y aplicaciones web modernos basados en la nube.

- **Compilar aplicaciones y juegos multiplataforma:** Visual Studio puede crear aplicaciones y juegos para macOS, Linux y Windows, así como para Android, iOS y otros dispositivos móviles. Con Visual Studio, puede crear:

  - Aplicaciones de .NET Core que se ejecutan en Windows, macOS y Linux.
  - Aplicaciones móviles para iOS, Android y Windows en C# y F# medianteXamarin.
  - Juegos 2D y 3D en C# mediante Visual Studio Tools para Unity.
  - Aplicaciones de C++ nativas para dispositivos iOS, Android y Windows. Comparta código común en bibliotecas para iOS, Android y Windows mediante C++ para desarrollo multiplataforma.

- **Conectarse a bases de datos:** El Explorador de servidores ayuda a explorar y administrar instancias y recursos de servidor de forma local y remota, y en Azure, Microsoft 365, Salesforce.com y sitios web.

  El Explorador de objetos de SQL Server ofrece una vista de los objetos de base de datos similar a la de SQL Server Management Studio. Con el Explorador de objetos de SQL Server puede realizar trabajos de administración y diseño de bases de datos ligeras. Algunos ejemplos son la edición de datos de tabla, la comparación de esquemas y la ejecución de consultas mediante menús contextuales.

- **Depuración y pruebas:** Con el sistema de depuración de Visual Studio, es posible depurar el código que se ejecuta en el proyecto

local, en un dispositivo remoto o en un emulador de dispositivo. Es posible ejecutar el código una instrucción cada vez, inspeccionandp las variables mientras se avanza. O bien, se pueden establecer puntos de interrupción que solo se alcanzan cuando una condición especificada es verdadera. Se pueden administrar las opciones de depuración en el propio editor de código para que no tenga que salir del código.Visual Studio ofrece opciones de prueba, como pruebas unitarias, Live Unit Testing, IntelliTest y pruebas de carga y rendimiento. Visual Studio también cuenta con funciones avanzadas de análisis de código para detectar errores de diseño, de seguridad y de otro tipo.

- **Implementación de la aplicación finalizada:** Visual Studio dispone de herramientas para implementar las aplicaciones en usuarios o clientes mediante Microsoft Store, un sitio de SharePoint o las tecnologías de InstallShield o Windows Installer.

- **Administrar el código fuente:** En Visual Studio, se puede administrar el código fuente en los repositorios de Git hospedados por cualquier proveedor, incluido GitHub. También puede buscar una instancia de Azure DevOps Server a la que conectarse.

Android Studio

Para el desarrollo del lado del cliente se ha decidido utilizar Android Studio, el cual es el entorno de desarrollo integrado oficial de Google para aplicaciones en Android. Desde el 7 de marzo del 2019 Kotlin es el lenguaje de programación preferido de Google para el desarrollo de aplicaciones en Android, aunque esta IDE también permita la implementación de las mismas en el lenguaje Java. [**?**]

Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas GNU/Linux, macOS, Microsoft Windows y Chrome OS. Ha sido diseñado específicamente para el desarrollo de Android.

Como lenguaje de programación se ha utilizado Java, ya que es un lenguaje que se ha utilizado a lo largo de la carrera en diferentes asignaturas, siendo uno de los lenguajes de programación más utilizados en los últimos años. Para el desarrollo del lado del cliente se ha optado por este entorno de desarrollo debido a que ya se sabía manejar de la asignatura de *Interacción Hombre-Máquina* del cuarto semestre de este grado, aun sabiendo que se tenía la opción de utilizar el mismo entorno que en el lado del servidor.

Las características de la versión más reciente de Android Studio a fecha de la entrega de segunda convocatoria, teniendo en cuenta que siempre se añaden nuevas funcionalidades en cada una de sus versiones, son las siguientes:[10]

- El soporte para la construcción de las aplicaciones está basado en Gradle, el cual ayuda a automatizar y administrar el proceso de compilación de las mismas mediante las dependencias que va añadiendo el usuario.

- La refactorización del código y de su estructura es específica de Android, teniendo también la posibilidad de realizar arreglos rápidos.

- Posee también herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas. Dichas herramientas han sido de gran utilidad para poder detectar los diferentes errores que han impedido que la aplicación se mostrase de la manera adecuada

- Integración de ProGuard y funciones de firma de aplicaciones. ProGuard es utilizado para la reducción y optimización del código de la aplicación del código, con la finalidad de que el rendimiento sea óptimo en los dispositivos móviles en los que se ejecuta la aplicación.

- Android Studio cuenta también con diferentes plantillas para crear diseños comunes de Android y otros componentes, pudiendo modificarse mediante un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario. Esta característica es fundamental para ayudar al desarrollador a elegir los mejores diseños base para las diferentes actividades de su aplicación, por lo que no es necesario tener amplios conocimientos en el lenguaje XML para empezar a desarrollarla.

- Android Studio también tiene soporte para programar aplicaciones para diferentes dispositivos, entre los cuales destacamos los teléfonos móviles, las tabletas, las aplicaciones de escritorio y los dispositivos de Android Wear.

- Android Studio tiene soporte integrado para Google Cloud Platform, que permite la integración con Firebase Cloud Messaging (antes 'Google Cloud Messaging') y Google App Engine.

- Para realizar las pruebas de la aplicación se cuenta con un dispositivo virtual de Android, teniendo también el soporte para la depuración inalámbrica para dispositivos físicos.
- El renderizado se realiza en tiempo real.
- Android Studio tiene su propia consola de desarrollador, además de tener la capacidad de integrar diferentes terminales dependiendo del sistema operativo que se esté utilizando.

Apoyándonos en las características anteriormente mencionadas, las principales ventajas de utilizar Android Studio son las siguientes:

- Como se ha mencionado con anterioridad, es la IDE oficial de Google para el desarrollo de aplicaciones de Android, desbancando a Eclipse en el año 2013.
- Permite la conversión de código Java a código Kotlin, algo que es imposible en otras IDEs como Eclipse, ya que Kotlin es un lenguaje el cual se ejecuta sobre una máquina virtual de Java, permitiendo también utilizar sus librerías.
- Permite programar la interfaz de la aplicación de forma interactiva, todo ello mediante los ficheros .xml del directorio /res/layout, pudiendo intercalar de forma sencilla entre la forma interactiva y el código .xml.
- Permite simular el funcionamiento de la aplicación sobre diferentes dispositivos, ya sean virtuales mediante su emulador, o físicos pudiendo conectar diferentes dispositivos mediante las opciones de desarrollador de los dispositivos Android.
- Permite inicializar proyectos a partir de plantillas preestablecidas, siendo de gran utilidad tanto para principiantes como para expertos.
- Permite la creación de módulos de Java, no sólo de módulos de Android, permitiendo así ejecutar esos módulos a parte para el desarrollo de diferentes pruebas para la versión básica de la ejecución del código de indicadores en línea de comandos.

En contraparte, los principales inconvenientes de Android Studio son los siguientes:

- Android Studio dificulta mucho la unificación del desarrollo de aplicaciones cliente-servidor bajo un mismo entorno, ya que resulta muy tedioso ejecutar tanto la aplicación como el servidor en dos hilos diferentes.

- Android Studio no soporta otros lenguajes de programación diferentes de Java y Kotlin, por lo tanto no se puede programar el servidor de *ASP.NET* en C#, obligando al uso de otra IDE distinta para su implementación, como *Visual Studio 2022*.

- El emulador de Android Studio en ocasiones tiene un desempeño que deja mucho que desear debido a su inestabilidad en tiempo de ejecución, sucediendo lo mismo con el desarrollo de los layout de las actividades, que en los modos *Split* y *Design* tiene diferentes problemas de renderización. Afortunadamente estos problemas no se han trasladado a los dispositivos físicos en los que se han realizado las pruebas, solventando el pobre desempeño que pueda tener el emulador.

- Android Studio no tiene un soporte nativo de Azure debido a que Android Studio es de Google y Azure es de Microsoft. Eso obliga al usuario a tener que buscar otro entorno diferenciado para poder realizar la implementación del lado del servidor, algo que se tuvo que hacer casi al final del tiempo de desarrollo del mismo, cuando se pasó de tratar de implementar un servidor embedido en la aplicación para la realización de preuebas del servidor en local, obligando a hacer este cambio para avanzar con el desarrollo

4.3. Lenguajes de programación y herramientas utilizadas

Lado del cliente

Java

En el lado del cliente se ha utilizado Java como lenguaje de programación debido a que, como se ha mencionado en la sección anterior, dicho lenguaje ya se había utilizado en las asignaturas de Metodología de la Programación, Estructuras de Datos, Interacción Hombre-Máquina, Programación Concurrente y de Tiempo Real, Aplicaciones de Bases de Datos, Testes e Qualidade de Software (equivalente en el *Instituto Superior de Engenharia de Coimbra* a la asignatura Validación de Datos de esta universidad) y Sistemas

Distribuidos, por lo que esta decisión viene respaldada por la experiencia otorgada por los docentes de dichas asignaturas durante todo el grado.
Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.
El lenguaje de programación Java fue desarrollado originalmente por James Gosling, de Sun Microsystems (en la actualidad propiedad de Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.
Por lo tanto, para el lado del cliente se han utilizado las siguientes herramientas de Java:

- *ASyncTask:* *ASyncTask* es una herramienta que se utiliza en el código para realizar las peticiones al servidor desde el cliente en segundo plano. A pesar de que lleva depreciada desde la API 30, sigue siendo una herramienta de gran utilidad para realizar estas peticiones a la base de datos y para obtener la respuesta a las mismas, por lo que supone una gran ayuda para la comunicación con el servidor de *Azure*. Entre las herramientas similares a *ASyncTask* se podía haber utilizado el paquete *java.util.concurrent* y *call* de *Retrofit* (que sí que se utiliza para construir el enlace base para la comunicación del cliente). El motivo por el cual se ha optado por utilizar ASyncTask es por ser una herramienta específica de Android y por la posibilidad que tiene de devolver los resultados de las operaciones, algo que es más complicado en las alternativas anteriormente mencionadas. *ASyncTask* está diseñado para ser una clase auxiliar que no constituye un marco genérico de subprocesos. Idealmente, AsyncTask debe usarse para operaciones cortas (unos pocos segundos como máximo). Si necesita mantener subprocesos en ejecución durante largos períodos de tiempo, se recomienda encarecidamente que use las diversas API proporcionadas por el paquete java.util.concurrent, como Executor, ThreadPoolExecutor y FutureTask, lo cual no se ha llegado a necesitar en este caso. La tarea asincrónica se define mediante un cálculo que se ejecuta en un subproceso en segundo plano y cuyo resultado se publica en el subproceso de la interfaz de usuario. Una tarea asíncrona se define por 3 tipos genéricos, llamados `Params`, `Progress` y `Result`, y 4 pasos, llamados `onPreExecute`, `doInBackground`, `onProgressUpdate` y `onPostExecute` (sólo se han utilizado `doInBackground` y `onPostExecute`).

Aquí se muestra un pequeño ejemplo sobre el uso de *ASyncTask*:

```
1         private class DownloadFilesTask extends
             ↪ AsyncTask<URL, Integer, Long> {
2             protected Long doInBackground(URL...
                 ↪ urls) {
3                 int count = urls.length;
4                 long totalSize = 0;
5                 for (int i = 0; i < count; i++) {
6                     totalSize += Downloader.
                         ↪ downloadFile(urls[i]);
7                     publishProgress((int) ((i / (
                         ↪ float) count) * 100));
8                     // Escape early if cancel() is
                         ↪ called
9                     if (isCancelled()) break;
10                }
11                return totalSize;
12            }
13
14            protected void onProgressUpdate(Integer
                 ↪ ... progress) {
15                setProgressPercent(progress[0]);
16            }
17
18            protected void onPostExecute(Long
                 ↪ result) {
19                showDialog("Downloaded " + result +
                     ↪  " bytes");
20            }
21        }
```

Como se puede comprobar, en primer lugar se van descargando los ficheros en el `doInBackground`, los cuales se obtienen del parámetro `Params` que es URL, luego el progreso se va estableciendo en el `onProgressUpdate` a medida que van avanzando las descargas y, por último, muestra un mensaje en el `onPostExecute` en el que se muestra el número de bytes descargados, como tercer parámetro `Result`.

- *OKHttp:* OkHttp es un cliente HTTP que es eficiente por defecto, ya que:
  - El soporte de HTTP/2 permite que todas las solicitudes al mismo servidor compartan un socket.

- La agrupación de conexiones reduce la latencia de las solicitudes (si no está disponible HTTP/2).
- La compresión transparente GZIP reduce el tamaño de las descargas.
- La caché de respuestas evita completamente la red en las solicitudes repetidas.
- OkHttp persevera cuando la red tiene problemas: se recuperará silenciosamente de problemas de conexión comunes. Si tu servicio tiene múltiples direcciones IP, OkHttp intentará con direcciones alternativas si la primera conexión falla. Esto es necesario para IPv4+IPv6 y servicios alojados en centros de datos redundantes. OkHttp admite funciones TLS modernas (TLS 1.3, ALPN, verificación de certificado). Se puede configurar para que tenga una conexión alternativa para una amplia conectividad.

Usar OkHttp es fácil. Su API de solicitud/respuesta está diseñada con constructores fluidos e inmutabilidad. Admite tanto llamadas de bloqueo síncronas como llamadas asíncronas con devoluciones de llamada.

En este caso *OKHttp* se utiliza en conjunto con *Retrofit* para la construcción de un cliente nuevo mediante la función `OkHttpClient().newBuilder().build()`:

```
client=new OkHttpClient().newBuilder().
    ↪ build();
```

- *Retrofit: Retrofit* es la clase a través de la cual las interfaces de API se convierten en objetos invocables. Por defecto, *Retrofit* proporciona valores predeterminados sensatos, pero también permite personalización. Por defecto, Retrofit solo puede deserializar cuerpos HTTP en el tipo `ResponseBody` de OkHttp y solo puede aceptar su tipo RequestBody para la anotación `@Body`. Por lo tanto, un ejemplo de Retrofit es el siguiente:

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.github.com/")
        .addConverterFactory(
            ↪ GsonConverterFactory.create()
            ↪ )
        .build();
```

Como se ha mencionado con anterioridad, *Retrofit* se utiliza en conjunto con *OkHttp* para poder enviar las peticiones al servidor y poder

recibir posteriormente sus respuestas. *Retrofit* proporciona las etiquetas necesarias para indicar a las APIs el tipo de consulta a realizar `@GET, @POST, @PUT y @DELETE`, el cuerpo a enviar junto con la solicitud `@Body` y los atributos a añadir al path `@Path`.

Lado del servidor

C#

Para el lado del servidor se ha decidido utilizar C# como lenguaje de programación, aunque en la gran mayoría de la fase de desarrollo se haya pretendido utilizar el mismo lenguaje para implementar toda la aplicación cliente servidor, utilizando JDBC para la conexión de la base de datos y JAX-RS como API para la aplicación web, se ha optado finalmente por C# junto con el framework de ASP.NET debido a que dicho lenguaje y dicho framework tienen el soporte integrado en *Azure* para la implementación y posterior despliegue de la aplicación web que soporta la base de datos, lo que ha hecho que los tiempos de respuesta de las solicitudes sean bastante más cortos en comparación con la alternativa anteriormente mencionada basada en Java.

C# es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Su sintaxis básica está basada en C y C++, utilizando también el modelo de objetos de la plataforma .NET, similar al de Java, lo que ha favorecido al rápido aprendizaje de este lenguaje.[11]

[4] *ASP.NET Core* es un marco multiplataforma de código abierto y de alto rendimiento cuyo fin es compilar aplicaciones que se encuentran en internet o en la nube. Este marco da la posibilidad de :

- Compilar servicios y aplicaciones web, aplicaciones de Internet de las cosas (IoT) y back-ends móviles.
- Efectuar implementaciones locales y en la nube.
- Ejecutar en .NET Core.

Las principales ventajas que tiene *ASP.NET* con respecto a su competencia son las siguientes:

- Da la posibilidad de crear una aplicación cliente-servidor de forma unificada, aunque este no haya sido el caso.
- Está diseñado para realizar pruebas
- ASP.NET dispone de las Razor Pages, que se trata un modelo de programación basado en páginas que facilita la compilación de interfaces de usuario web y hace que sea más productiva.
- Blazor permite usar C# en el explorador, junto con JavaScript, permitiendo compartir la lógica entre el cliente y el servidor.
- Capacidad para desarrollarse y ejecutarse en cualquier sistema operativo.
- De código abierto y centrado en la comunidad.
- Integración de marcos del lado cliente modernos y flujos de trabajo de desarrollo.
- Compatibilidad con el hospedaje de servicios de llamada a procedimiento remoto con gRPC.
- Un sistema de configuración basado en el entorno y preparado para la nube.
- Tiene la inserción de dependencias integrada.
- Tiene una canalización de solicitudes HTTP ligera, modular y de alto rendimiento.
- Tiene la capacidad de hospedar diferentes tipos de servidores.
- Control de versiones en paralelo.
- Herramientas que simplifican el desarrollo web moderno.

Azure SQL

[7] *Azure SQL Database* es un motor de base de datos de plataforma como servicio (PaaS) totalmente administrado que se encarga de la mayoría de las funciones de administración de bases de datos, incluidas *la supervisión sin intervención del usuario, la aplicación de revisiones, la creación de copias de seguridad y la actualización*. El motor de base de datos de Azure SQL Server se ejecuta siempre en la versión más reciente y estable del motor de

base de datos de SQL Server, así como en un sistema operativo revisado que tiene una disponibilidad del 99,99 %. Las funcionalidades de PaaS en Azure SQL Database permiten concentrarse en las actividades de administración y optimización de bases de datos específicas del dominio que son importantes para el negocio.

El motor de base de datos más reciente de *Microsoft SQL Server* es la base de *Azure SQL Database*, permitiendo utilizar características avanzadas de procesamiento de consultas, como el procesamiento de consultas inteligente y las tecnologías de memoria de alto rendimiento. De hecho, las últimas funcionalidades de SQL Server se publican primero en Azure SQL Database y luego en SQL Server mismo. Las funcionalidades más recientes de SQL Server se pueden obtener sin costo mediante actualizaciones o revisiones, y se han probado en millones de bases de datos.

En cuanto a las opciones de implementación de base de datos de Azure SQL, podemos resaltar dos:

- **Una base de datos única** es una base de datos aislada que se administra completamente. Si tiene aplicaciones y microservicios modernos en la nube que necesitan un solo origen de datos confiable, puede usar esta opción. Una sola base de datos es similar al motor de base de datos de SQL Server. Es la opción más sencilla para nuestro caso, sobre todo al principio del desarrollo.
- El **grupo elástico** es una colección de bases de datos distintas con un conjunto compartido de recursos, como la memoria o la CPU. Un grupo elástico puede permitir el movimiento de una sola base de datos. Éste no se ha implementado en esta aplicación, aunque

Aspectos relevantes del desarrollo del proyecto

5.1. Ciclo de vida utilizado

El ciclo de vida utilizado para el desarrollo de esta aplicación, que se ha procurado seguir durante todo el tiempo de desarrollo, es un ciclo de vida de tipo ágil. Se ha escogido este tipo de ciclo de vida puesto que permite tener el control de todos los aspectos de las diferentes fases de desarrollo de la aplicación, tanto a nivel de desarrollo como a nivel de documentación, permitiendo también la alternancia entre fases acorde con las necesidades del proyecto.

Un ciclo de vida ágil consiste en una entrega de forma incremental de las diferentes características de la aplicación y mejora de las características ya existentes de la misma. Durante este ciclo de vida se han ido subiendo diferentes modificaciones tanto a *GitHub* como a *Trello*, las cuales son denominadas como sprints. Cada sprint representa una actualización de una característica o conjunto de características de la aplicación o de los anexos, las cuales ayudan a marcar qué actividad se ha realizado y en qué momento se ha realizado.
En este caso, se ha utilizado un método visual de Kanban.[2]

Kanban es un método de gestión del flujo de trabajo para definir, gestionar y mejorar los servicios que proporciona el trabajo de conocimiento, siendo de gran ayuda para la visualización del trabajo, la optimización de la eficiencia y la continuación de diferentes mejoras. El trabajo se representa en tableros Kanban, lo que permite optimizar la entrega de trabajo a través

de múltiples equipos y manejar, incluso los proyectos más complejos en un solo entorno.

La implementación de este modelo de ciclo de vida el ha sido posible gracias a *Trello*, que es una herramienta muy útil para la gestión de proyectos a nivel visual en el que cual se tienen diferentes columnas sobre las actividades por hacer, las actividades en realización, las actividades realizadas y las diferentes reuniones de seguimiento que se han tenido con el tutor.

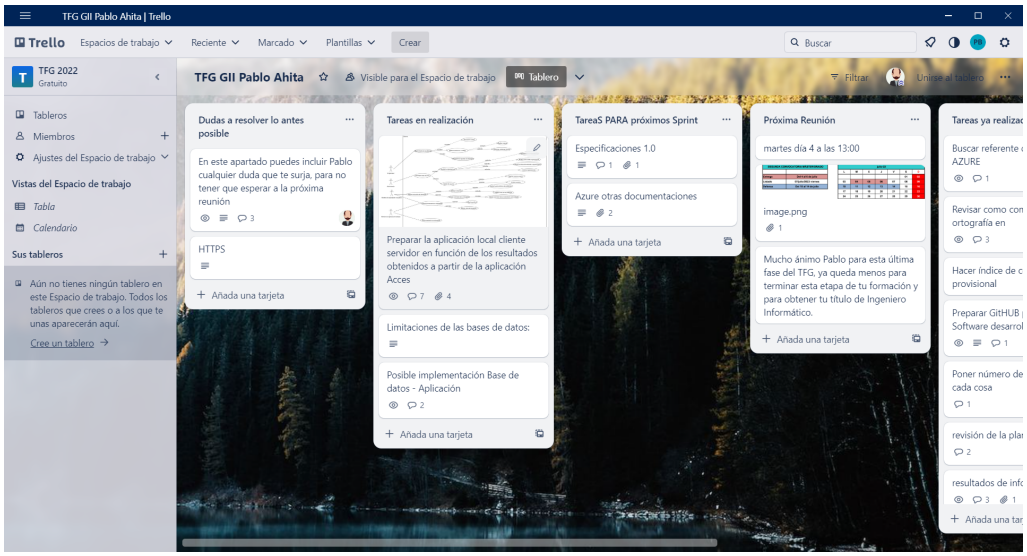

Figura 5.1: Uso de *Trello* para el proyecto

Como se puede comprobar en la siguiente captura, durante todo este tiempo se han colocado diferentes columnas en las que se han colocado diferentes aspectos:

- **Dudas a resolver lo antes posible:** En este apartado el alumno tiene la posibilidad de colocar esas inquietudes que tiene en cualquier aspecto relacionado con el proyecto sin necesidad de comentarlo explícitamente en cualquiera de las reuniones de seguimiento que se han ido realizando durante el curso.
- **Tareas en realización:** En este apartado se introducen las tareas en realización en ese mismo momento, las cuales se han ido marcando en cada una de las reuniones de seguimiento a lo largo de todo el tiempo de desarrollo.

- **Tareas para próximos sprint:** En este apartado se introducen las tareas que se realizarán para próximos sprints.
- **Próxima reunión:** En este apartado se introduce la información sobre la fecha de la próxima reunión y a la hora en la que tendrá lugar. Durante todo este tiempo las reuniones han tenido lugar los martes de cada semana por la mañana, ya sea a las 11:30 horas o a las 13:00 horas.
- **Tareas ya realizadas:** En este apartado se introducen todas las tareas que ya han sido realizadas durante el tiempo de desarrollo.
- **Información de las reuniones realizadas:** En este apartado se registran todas las reuniones realizadas durante todo este tiempo. Todas ellas han estado en el apartado anterior.

5.2. Fases de análisis, diseño e implementación

Para poder seguir el ciclo de vida anteriormente mencionado, se tienen que tener claras las fases a seguir durante todo el proceso de desarrollo. Al tratarse de un ciclo de vida ágil, cabe resaltar que el paso entre fases es cíclico puesto que, como se ha mencionado con anterioridad, conviene que el proceso de desarrollo de la aplicación se adapte a las necesidades que se vayan teniendo a lo largo del proyecto, por lo que las fases de *análisis*, *diseño* e *implementación* se van alternando a lo largo del tiempo.

Análisis

El análisis del entorno ha sido realizado con la ayuda principal de la aplicación en Microsoft Access, la cual ha servido como punto de referencia para montar las bases en las que se apoya dicha aplicación.
En la aplicación Access ya había posibilidad de registrar organizaciones, de crear equipos evaluadores, de realizar test de indicadores y de guardar los registros de dichos test de indicadores en la base de datos en este mismo programa. Era un procedimiento bastante arcaico el uso de esa aplicación, por lo que se ha tenido que plantear que nuevas características se pretenden añadir en esta nueva versión:

- En primer lugar, la aplicación tiene que almacenar los usuarios mediante el registro de los mismos en la base datos, en lugar de hacerlo

directamente en el apartado de equipos evaluadores. Los usuarios se segregan en tres categorías diferentes:

- **Usuarios de la Fundación Miradas o usuarios de organización evaluadora:** Estos usuarios tienen la capacidad de crear equipos evaluadores, de añadir organizaciones nuevas y centros nuevos, y de poder crear y visualizar evaluaciones de indicadores.
- **Usuarios de organización evaluada:** Estos usuarios pertenecen a las organizaciones que participan en estos test de indicadores junto con la *Fundación Miradas*. Dichos usuarios tienen la capacidad de ver los resultados de los diferentes equipos evaluadores
- **Administrador:** Es el encargado de gestionar las tablas de la base de datos, ya sea creando nuevas, eliminando existentes y modificando las columnas de las mismas. Es el único de los tres tipos de usuarios que no se puede registrar de la forma habitual.

Se ha decidido segregar de esta forma a los usuarios por pura distinción de sus responsabilidades (*Para ver más detalles ir al capítulo de Requisitos en los anexos*), algo que en la aplicación antigua de Access no se tenía o no se necesitaba tener en cuenta en su momento.

- Al igual que los usuarios, la segregación de organizaciones es también un factor de vital importancia. Dicha segregación es en la que se basan los usuarios para determinar si el usuario es de la *Fundación Miradas* o por lo contrario si es un usuario de otra organización. Cada una de las organizaciones se identifican por **identificador de organización** de tipo entero, un **tipo de organización** que puede ser *EVALUATED* o *EVALUATOR* y una **enfermedad**, que en nuestro caso siempre es *AUTISM*. Se ha decidido que se identifiquen de esta manera para que esta entidad pueda ser utilizada también para otro tipo de diagnósticos o enfermedades, requisito exigido por el tribunal antes de comenzar con el desarrollo. Por lo tanto, nos podemos encontrar con dos grandes tipos de organizaciones:

  - **Organizaciones evaluadoras (*Fundación Miradas*):** Dichas organizaciones no pueden ser añadidas como tal desde la aplicación, ya que la propia *Fundación Miradas* es la encargada de añadir organizaciones evaluadas, como se ha relatado con anterioridad. Para poder añadir una se tendría que utilizar *cURL* o *Azure Studio*. También tiene la potestad de añadir nuevos equivos evaluadores y de gestionar los test de indicadores.

  - **Organizaciones evaluadas:** Son las organizaciones que se someten a los diferentes test de indicadores. En contraparte con el otro tipo de organizaciones, éstas sí que pueden ser añadidas a partir de la propia aplicación, mediante el formulario de agregación de organizaciones. (*Véase manual de usuario en los Anexos o vídeos incluidos en el repositorio*)

- También se tiene que tener el control de los equipos evaluadores por parte exclusiva de los usuarios de la *Fundación Miradas*, a diferencia de la aplicación en Access la cual no hace distinción, como se ha mencionado con anterioridad. La estructura de los equipos evaluadores no ha cambiado con respecto a Access.

- La realización de las pruebas de indicadores también sigue la misma dinámica, con la diferencia de que la realización de los mismos es más intuitiva y rápida gracias al uso en pantallas táctiles.

Teniendo en cuenta esos aspectos, se ha realizado tanto el análisis de los casos de uso, el cual se encuentra en el capítulo de requisitos de los anexos, como el diagrama de clases.
En cuanto a la toma de decisiones, al principio del desarrollo se pretendía unificar todo bajo el mismo lenguaje *Java*, ya que cuento con la experiencia en este lenguaje que se ha ido proporcionando a lo largo de las asignaturas del grado. Para la comunicación con la base de datos se utilizaba *JDBC*, el cual ejecuta las consultas SQL desde el propio código. Durante gran parte del tiempo de desarrollo de la aplicación, se han ido encontrando diferentes dificultades para poder ejecutar dichas órdenes en la base de datos, puesto que todo aquello que requiera una conexión a internet desde una aplicación Android, requiere que sea ejecutado en segundo plano, ya sea mediante *ASyncTask*, mediante callbacks o mediante técnicas de programación concurrente. Ese inconveniente obligó a empezar a trabajar con las operaciones al servidor en segundo plano, puesto que en un hilo tiene que estar ejecutándose la aplicación y en otro tiene que estar ejecutándose la solicitud al servidor. También se pretendía montar el servicio web en *JAX-RS* para que pueda realizar esas operaciones en la base de datos mediante *JDBC*. Aunque esta técnica funcionaba correctamente, el procesamiento de la respuesta en Azure era bastante más lento debido a que no está integrado en Azure, en comparación a *C#* y al framework *ASP.NET*, los cuales por su integración en Azure tiene unos tiempos de respuesta bastante más rápidos en comparación con su semejante en *Java*.

Diseño

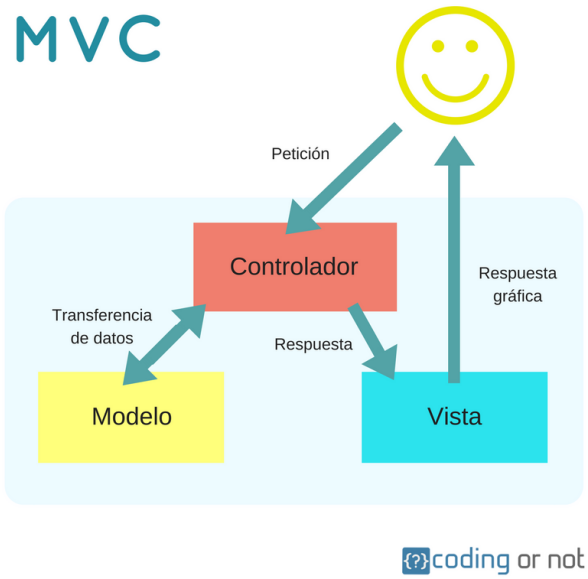

Figura 5.2: Estructura básica de la arquitectura *Modelo-Vista-Controlador*

En este caso se ha elegido como arquitectura el modelo denominado *Modelo-Vista-Controlador*. Dicho modelo segrega las diferentes responsabilidades que tienen los usuarios en los que cada una de las tres capas (*Modelo, Vista y Controlador*) tiene unas reglas exclusivas de ellas mismas.

- **Modelo:** El modelo se encarga de almacenar las diferentes entidades con las que interactúa el servidor, para poder realizar así las peticiones y poder devolver correctamente la respuesta al cliente, previa serialización en el cliente y previa deserialización en el servidor. En este caso el modelo consta de las clases *User, Organization, Indicator, Evidence, Center, Address, City, Province, Region, Country, EvaluatorTeam, EvaluatorTeamMember, IndicatorsEvaluation* e *IndicatorsEvaluationReg* (*Véase más información acerca de los modelos en el capítulo de Diseño de los Anexos*). Cabe resaltar también que el modelo tiene que ser idéntico tanto en el cliente como en el servidor, para evitar así diferentes problemas en la serialización y deserialización de dichas entidades.

- **Vista:** La vista consiste en la forma en la que se muestran el contenido de las respuestas de las peticiones en la base de datos. También se

incluye en este apartado la interfaz gráfica con la que interactúan los usuarios. En este caso, como cada tipo de usuario tiene responsabilidades diferentes, se considera que esta aplicación tiene diferentes vistas, todo por las responsabilidades que tienen los usuarios. En nuestro caso, la vista es proporcionada al cliente mediante las clases denominadas *Caller*, las cuales se encargan de enviar la solicitud al endpoint correspondiente del servidor y de procesar correctamente la respuesta que éste le da.

- **Controlador:** El controlador se encarga de realizar las peticiones a la base de datos previa solicitud del cliente. El controlador puede realizar todo tipo de operaciones *CRUD (Create, Read, Update, Delete)* de la base de datos, encargándose también de marcar los endpoints en los que se van a realizar dichas operaciones. Dichos métodos del controlador se apoyan en los *Services* que realizan la consulta de la base de datos, todo ello en el lado del servidor.

Se ha optado por esta arquitectura debido a que se busca el control necesario para todo el proceso de las peticiones HTTP entre el cliente y el servidor. En primer lugar los modelos son los que se encargan de establecer la estructura JSON de las peticiones, de tal forma que en el proceso de serialización y de deserialización los campos sean asignados correctamente, para que la vista muestre las respuestas de la manera esperada de las peticiones de las operaciones realizadas por el controlador.

En cuanto a la base de datos se cuentan con las siguientes entidades SQL, se utiliza `CREATE TABLE` para poder crear las tablas, `INSERT INTO nombre_tabla VALUES` para modificar las tablas, `UPDATE nombre_tabla SET columna=valor WHERE columna=valor_old` para actualizar la columna, `SELECT * FROM nombre_tabla` para mostrar los registros de una tabla y `DELETE * FROM nombre_tabla` para hacer el delete de esa tabla. Para la información sobre la estructura de las entidades, *véase el capítulo de diseño de los anexos.*

Implementación

En cuanto a la implementación de la aplicación, se han ido encontrando diferentes problemas que se han ido solventando con el paso del tiempo, quedando algunas pendientes de resolver. Los errores que se han ido encontrando a lo largo del tiempo son los siguientes:

- `Android.NetworkOnMainThreadException:` Esta excepción ha saltado sobre todo al principio del proceso de desarrollo, en la que se estaba intentando conectar directamente la base de datos local a la aplicación sin intermediario alguno como el caso de un servidor. Esta excepción sale debido a que las operaciones de la base de datos y de comunicación pretenden hacerse en el mismo hilo en el que se ejecuta la aplicación, haciendo que esta se cierre repentinamente por esa incorrecta gestión de hilos. Para evitar que salte esta excepción en el cliente, se tiene que montar esa estructura *Modelo-Vista-Controlador* en el cliente vista con anterioridad, utilizando cualquier técnica de manejo de hilos, decantándome por *ASyncTask* por la razón mencionada en el capítulo anterior, donde se menciona la herramienta y las posibles alternativas a la misma.

- `400 Bad Request:` Esta excepción ha saltado debido a que el JSON no ha sido generado correctamente. Este fallo surge debido a un problema en la estructura del modelo o la serialización en el lado del cliente, ya que la forma en la que se envía el JSON no depende como tal del lado del servidor, que puede estar funcionando correctamente. Para evitar que salga este código de error, hay que asegurarse que en el cliente los campos se llamen exactamente igual que en el JSON que recibe el cliente, al igual que en el servidor y en la propia base de datos.

- 500 Internal Server Error: Esta excepción surge por un problema interno en el servidor, por lo que el cliente no tiene por qué tener fallos en el modelo o en el procesamiento de la respuesta. Este fallo puede surgir por diferentes razones, por lo que es buena práctica hacer un debug en local del lado del servidor mediante una llamada al endpoint desde herramientas como *cURL*. Ejemplo:

```
curl -X POST -H "Content-Type: application/
   ↪ json" -d "{\"idOrganization\":1,\"
   ↪ orgType\":\"EVALUATED\",\"illness
   ↪ \":\"AUTISM\",\"idCenter\":1,\"
   ↪ centerDescription\":\"Sede principal
   ↪ \",\"idAddress\":2,\"telephone
   ↪ \":987654321}" https://localhost
   ↪ :7049/Centers
```

  Hay que considerar también que los puntos clave o breakpoints tienen que estar activados según la conveniencia que tenga el usuario en ese momento, así se puede observar paso a paso el motivo los pasos a seguir

hasta llegar a la zona crítica. Este error puede surgir por sintaxis en la orden SQL y por violaciones de claves primarias y foráneas, al igual que por cualquier tipo de error que surja en el servidor.

Internacionalización

La internacionalización de una aplicación es un aspecto clave para garantizar su accesibilidad y usabilidad en diferentes países y culturas. En el caso de una aplicación de Android Studio, es posible implementar la internacionalización de manera efectiva utilizando diferentes recursos proporcionados por la plataforma. A continuación, se presenta un apartado sobre la internacionalización de la aplicación, considerando los idiomas español, inglés y francés, y la posibilidad de ampliarlo a otros idiomas en el futuro:

- **Internacionalización de la interfaz gráfica:** La interfaz gráfica de la aplicación se ha traducido a tres idiomas: español, inglés y francés. Android Studio proporciona una herramienta llamada *Translations Editor* que facilita la gestión de los ficheros strings.xml para cada idioma. Esta herramienta permite configurar de manera interactiva los textos de la interfaz en los diferentes idiomas. ara implementar la internacionalización de la interfaz gráfica en la aplicación, se deben seguir los siguientes pasos:

  - *Creación de ficheros strings.xml:* Para cada idioma, se deben crear ficheros strings.xml separados en diferentes directorios. Por ejemplo: strings.xml para el idioma por defecto (inglés) en el directorio *values*, strings.xml para el español en el directorio *values-es* y strings.xml para el francés en el directorio *strings-fr*.
  - *Definición de cadenas de texto:* En cada fichero strings.xml, se deben definir las cadenas de texto utilizadas en la interfaz gráfica de la aplicación. Cada cadena debe tener un identificador único y su correspondiente traducción en el idioma correspondiente.
  - *Acceso a las cadenas de texto en el código:* Para mostrar las cadenas de texto en la interfaz, se debe acceder a ellas mediante su identificador en el código de la aplicación. Por ejemplo, utilizando `getString(R.string.mi_cadena)`.

  Con estos pasos, la interfaz gráfica de la aplicación estará internacionalizada y mostrará los textos correspondientes al idioma configurado en el dispositivo siempre y cuando éste esté disponible.

- **Internacionalización de los datos obtenidos de la base de datos:** Además de la interfaz gráfica, es importante internacionalizar los datos obtenidos de la base de datos de la aplicación. En este caso, se ha optado por utilizar la biblioteca Locale de Java para detectar el idioma del dispositivo y mostrar los indicadores y evidencias en el idioma correspondiente.Para implementar la internacionalización de los datos, se pueden seguir los siguientes pasos:

  - *Obtención del idioma del dispositivo:* Utilizando la clase `Locale` de Java, se puede obtener el idioma configurado en el dispositivo. Por ejemplo: `Locale.getDefault().getLanguage()`.
  - *Obtención de los datos de la base de datos:* Al obtener los datos de indicadores y evidencias de la base de datos, se deben considerar las traducciones correspondientes para cada idioma. Por ejemplo, si el idioma del dispositivo es español, se deben obtener los datos en español; si es francés, se deben obtener en francés; de lo contrario, se obtienen en inglés.
  - *Visualización de los datos en la interfaz:* Una vez obtenidos los datos en el idioma correspondiente, se pueden mostrar enla interfaz de la aplicación según el diseño y la estructura definida.

  Es importante tener en cuenta que la implementación de la internacionalización para los datos de la base de datos puede variar según la estructura y la forma en que se accede a los datos en la aplicación. La biblioteca *Locale* de Java proporciona diferentes métodos y opciones para adaptar la aplicación al idioma del dispositivo.

Trabajos relacionados

Los trabajos final de grado de referencia que se han utilizado son los siguientes:

- **Prueba de concepto *Azure Monitor***: Es un Trabajo Final de Grado que trata sobre aplicaciones APM (*Application Performance Monitor*), que son herramientas que diagnostican el rendimiento de las aplicaciones, con la finalidad de encontrar fallos en el programa, cuellos de botella , e incluso poder solucionarlos, evitando degradaciones. La funcionalidad utilizada en este TFG es *Azure Monitor*, el cual es la herramienta de este estilo proporcionada por Microsoft, cuyo objetivo es averiguar si el proyecto puede ser utilizado por una organización que tiene dos entornos completamente diferentes: un entorno de nube *Azure* donde se encuentra su sitio web y que podrá albergar en el futuro alguno de los servicios que ofrece, y otro entorno en el sitio donde residen el resto de los servicios.

- **Desarrollo de una plataforma de tratamiento y streaming de vídeo para difusión de la cultura utilizando instancias de *Azure***: Es un Trabajo Final de Grado que trata sobre una herramienta que es utilizada por artistas de cine independientes para darse a conocer a través de publicidad en diferentes dispositivos, ya sea a través de aplicaciones móviles o a través de navegadores. Azure es utilizado como soporte para la base de datos y la posterior gestión de transacciones, lo que se busca para el proyecto de la Fundación Miradas.

- **Servicios en la nube con *Microsoft Azure* : desarrollo y operación de una aplicación Android con DevOps**: Este proyecto

trata sobre el desarrollo de una aplicación de Android con almacenamiento en la nube, utilizándose para el almacenamiento, visualización y procesado de fotografías. Dicho proyecto utiliza *Azure* para desplegar la aplicación en la nube, con ayuda de DevOps para poder alargar el ciclo de vida de la misma. Puede ser de utilidad también para poder desarrollar la aplicación tanto a nivel de frontend como a nivel de backend.

- ***JIZT*. Generación de resúmenes abstractivos en la nube mediante Inteligencia Artificial**: JIZT es un servicio de generación automática de resúmenes basado en la corriente *Cloud Native*, que se basa en los principios de los sistemas escalables, elasticidad y agilidad. Dicho servicio es sustentado por una arquitectura de micro-servicios dirigido por eventos, garantizando la alta disponibilidad del servicio,aparte de los tres principios mencionados con anterioridad. Dicha aplicación es multiplataforma, por lo que consume la API REST del servicio en la nube, donde cualquier usuario dispone de los resúmenes que desee.

- **UBUNurse**: Este proyecto consiste en una aplicación multi-dispositivo el cual almacena en la nube registros sobre la evaluación de la atención domiciliaria por parte de dicho personal hacia un determinado paciente. El procedimiento a seguir consiste en elegir un paciente en la lista de la cual dispone el enfermero, posteriormente se elige el test a realizar y por último se obtienen los resultados de la realización de dicho test. Este software pretende automatizar dicho proceso para mejorar la eficiencia del personal sanitario y también para mejorar el proceso de evaluación de cada paciente.

- **Machine learning mediante *Microsoft Azure*: una aplicación sobre real-state**: En este proyecto se trata más a fondo las herramientas de las cuales dispone *Azure*, las cuales son utilizadas para la creación de elementos de machine learning, los cuales explican el funcionamiento de *Azure* con fines estadísticos. Se utiliza en dicho proyecto *Azure Machine Learning* con los datos empíricos de una inmobiliaria estadounidense para la predicción y posterior clasificación del valor de las viviendas, comparando posteriormente con otros modelos de clasificación y de regresión. A pesar de no ser un proyecto de bases de datos, se le da mucho hincapié a los gráficos y a la muestra de resultados, lo que puede servir de utilidad para la parte final de muestra de resultados.

- **Desarrollo de un Bot en la plataforma *Azure* para ayudar en el aprendizaje del lenguaje de programación C**: Este proyecto consiste en el desarrollo de un bot mediante *Azure* para ayudar al alumnado de 1º curso del Grado en Ingeniería Electrónica y Automática de la Universidad Politécnica de Cartagena al aprendizaje del lenguaje de programación C. Aparte de eso trata también sobre las herramientas de Inteligencia Artificial de *Azure*, como *LUIS (Language Understanding)*, aparte de que el software de dicho bot tiene soporte de incorporación de diferentes idiomas, lo que es un factor de gran importancia para el desarrollo de la app. Se ha escogido este trabajo de final de grado puesto que se trata de un proyecto bastante completo en cuanto a contenido a sacar de él y en cuanto a estructuración del contenido del mismo.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Por lo tanto las posibles mejoras que se pueden aplicar a esta aplicación, podemos destacar las siguientes:

- En la pantalla de adición de organizaciones, cuando se añaden el director y el profesional de la organización evaluada, no se actualizan correctamente, correctamente los desplegables al retornar al formulario de la organización. El resto de campos sí se actualizan correctamente. Para ello, se tienen que modificar la lógica de los desplegables en el lado del cliente de tal forma que las opciones elegidas perduren.

- Se tiene que implementar un sistema de comprobación de credenciales para el inicio de sesión y para el registro de los usuarios, el cual envía un correo electrónico o un SMS con un código generado para garantizar la doble autenticación y a su vez la existencia de esas credenciales introducidas. Para ello se tiene que establecer un servidor SMTP y una cuenta de correo electrónico para poder generar dicho código de validación, pudiéndose hacer tanto en el lado del cliente como en el lado del servidor, preferiblemente en este último lado.

- Se tiene la idea también de desarrollar un algoritmo de aprendizaje supervisado para que a partir de los resultados de otros test de indicadores haga la predicción de cuáles van a ser los resultados obtenidos en el futuro, reforzando también la mejora de la muestra de resultados en el futuro. Pueden utilizarse para ello diferentes técnicas de aprendizaje supervisado, como es el caso de las redes neuronales con una técnica de aprendizaje hebbiano.

- En la pantalla de añadir equipos evaluadores no se añaden el consultor externo, el profesional de atención directa y el responsable como objetos de clase `EvaluatorTeamMember`. Para solucionarlo se tiene que modificar la lógica del lado del servidor para esta operación, ya que la causa es el error *500 - Internal Server Error*.

- Los centros no se pueden añadir de forma independiente, debido a un error 400 Bad Request. Seguramente se trate de algún problema en el cliente, puesto que haciendo `curl` de ese endpoint ejecuta sin problemas. Por lo tanto, hay que modificar la forma en la que la respuesta es generada y la forma en la que se deserializan los centros en el cliente.

- En la parte de actividad reciente, no se muestran los resultados de cada uno de los test de indicadores, aunque estos se puedan mostrar perfectamente a través de los endpoint correspondientes.

- Se va a desarrollar la generación de los gráficos de los resultados de los test de indicadores, disponiendo para ello de la estructura necesaria para ello, teniendo *Azure* diferentes recursos para la generación de los mismos.

- Se va a desarrollar una pantalla para que la propia *Fundación Miradas* pueda añadir, eliminar y modificar los indicadores y sus respectivas evidencias de igual forma que otras actividades de tipo formulario.

- Se va a implementar en *Azure* un despliegue de tipo dinámico, en el cual cada vez que se haga un commit en el repositorio de *GitHub*, los cambios en el despliegue se harán efectivos de forma automática.

- Se pretende mejorar la seguridad en la muestra de los endpoints, buscando la manera de que algunos campos sensibles sean invisibles a la hora de mostrar los endpoint. Se ha tratado de poner el campo `passwordUser` en el servidor como `internal`, el cual sólo es visible en el ensamblado, pero finalmente se desechó esa idea debido a que genera complicaciones entre cliente y servidor.

- En la versión final no se va a utilizar HTTP para la realización de las peticiones, buscando alternativas bastante más seguras, como puede ser el uso de *Entity Framework*, una ORM (Object-Relational Mapping) que puede realizar esas operaciones CRUD sin escribir las consultas SQL en el código del servidor.
- Se va a configurar un menú de opciones para que la aplicación pueda configurar la interfaz gráfica de la aplicación a su gusto y necesidades, incluyendo entre estas características el modo de lectura fácil para aquellas personas que lo necesitan.
- Se pretende a su vez también desarrollar esta aplicación también en forma de aplicación web y para *iOS*, ampliando sobremanera el marco de usuarios ya de por sí amplio con los usuarios de *Android*. Para ello puede utilizarse también herramientas de *Visual Studio 2022* y de *C#*, tal y como se menciona en el capítulo de *Técnicas y Herramientas* de esta memoria.
- Se pretende ampliar la internacionalización de la aplicación mediante las herramientas mencionadas en el capítulo de Aspectos relevantes del desarrollo

En cuanto al propio proyecto en sí, ha sido un proyecto que me ha llenado muchísimo, ya que por mi discapacidad cognitiva siempre he estado muy concientizado sobre todos los aspectos que las personas con cualquier discapacidad, en este caso las personas con TEA, por lo que es una aplicación hecha para ayudar a las personas con discapacidad por parte de una persona con una discapacidad reconocida.

A la par dicho proyecto ha supuesto un gran aprendizaje para mí en todos los aspectos. En primer lugar en el aspecto laboral o académico, ya que he tenido que refrescar conocimiento sobre algunas herramientas las cuales llevaba bastante tiempo sin utilizar y también he tenido que aprender a manejar herramientas las cuales desconocía en muy poco tiempo, como es el manejo de *Visual Studio 2022* con el lenguaje de programación *C#* y el framework *ASP.NET*, los cuales no se han impartido en las asignaturas de este grado, por lo que ha supuesto un aporte de conocimientos adicional para la entrada al mundo laboral. Posteriormente también está suponiendo un aprendizaje importante a nivel personal, ya que ha habido bastantes momentos difíciles producto de las diferentes dificultades que han ido surgiendo durante el tiempo de desarrollo del mismo los cuales han sido un desafío a nivel personal bastante grande, teniendo la fortuna de haber recibido

el apoyo de mucha gente, como menciono en la parte de agradecimientos de esta memoria, lo cual ha supuesto un impulso muy grande para seguir peleando para sacar este proyecto adelante de la mejor forma posible y a su vez afrontar los diferentes desafíos que un Ingeniero Informático debe cumplir en el día a día y afrontarlos de forma tranquila y serena.

Bibliografía

[1] Universidad de Burgos. Azure for education - universidad de burgos, 2020. [Página de acceso al enlace de Azure for Education].

[2] kanbanize.es. ¿qué es kanban? explicación para principiantes, 2023. [¿Qué es Kanban? Explicación para principiantes].

[3] Microsoft Learn. Características de visual studio, 2023. [Características de Visual Studio].

[4] Microsoft Learn. Información sobre asp.net core, 2023. [Información sobre ASP.NET Core].

[5] Microsoft Learn. ¿qué es visual studio?, 2023. [¿Qué es Visual Studio?].

[6] Microsoft. Azure for education, 2023. [Página principal de Azure for Education].

[7] Microsoft. Azure sql, 2023. [Azure SQL].

[8] Microsoft. Microsoft azure, 2023. [Página principal de Microsoft Azure].

[9] Microsoft. Microsoft learn, 2023. [Página oficial de Microsoft Learn].

[10] Wikipedia. Android studio, 2023. [Artículo de Wikipedia sobre Android Studio].

[11] Wikipedia. Java, 2023. [Artículo de Wikipedia sobre C Sharp].