FACULTAD DE INGENIERIA Y CIENCIAS APLICADAS

DESARROLLO DE UNA PLATAFORMA ORIENTADA PARA LA ADMINISTRACIÓN DE LOS TRATAMIENTOS Y ACTIVIDADES DE REHABILITACIÓN DE ADULTOS MAYORES

AUTOR

ANTONIO FRANCISCO LÓPEZ POSSO

AÑO

2019

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DESARROLLO DE UNA PLATAFORMA ORIENTADA PARA LA ADMINISTRACIÓN DE LOS TRATAMIENTOS Y ACTIVIDADES DE REHABILITACIÓN DE ADULTOS MAYORES

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor Guía

Msc. Ángel Gabriel Jaramillo Alcázar

Autor

Antonio Francisco López Posso

Año

2019

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Desarrollo de una plataforma orientada para la administración de los tratamientos y actividades de rehabilitación de adultos mayores, en el semestre 201920, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Ángel Gabriel Jaramillo Alcázar

Magister en Gerencia de Sistemas y Tecnologías de la Información

CC: 1715891964

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber dirigido el trabajo, Desarrollo de una plataforma orientada para la administración de los tratamientos y actividades de rehabilitación de adultos mayores, en el semestre 201920, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Luis Santiago Criollo Caizaguano

Master en redes de Comunicaciones

CC: 1717112955

DECLARACIÓN DE AUTORIA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

_____

Antonio Francisco López Posso

CC: 1002987855

AGRADECIMIENTOS

Agradezco a mis padres y hermanas por su apoyo y toda la confianza entregada durante el transcurso de mi vida académica, en especial en este proyecto.

A mis “abuelitas”, quienes fueron la fuente de inspiración de este trabajo. A mis profesores, por brindarme sus conocimientos y a mi querida universidad que ha el pilar de mi formación superior.

DEDICATORIA

Quiero dedicar este trabajo a mi familia, en especial a esos dos bellos seres: "Abuelita Bel" y "Abuelita Lucía", quienes motivaron la creación de este proyecto y que ahora son dos luceros en el cielo.

También, dedico mi trabajo, a todo colega que desee innovar, y generar un proyecto que permita mejorar la calidad de vida de las personas, en este caso, adultos mayores, un grupo que ha sido marginado tanto tiempo, rompiendo todo paradigma tecnológico que los rodea.

RESUMEN

El presente proyecto consiste en la implementación de herramientas tecnológicas, desarrolladas por el autor, para poder brindar una terapia, exclusivamente a un público de tercera edad (de 60 años en adelante). De tal manera que, de base fundamental, se implementó una aplicación desarrollada por el autor, que fue un proyecto final para una materia de la carrea, llamada "*Aplicaciones Móviles*", la misma que generó un gran impacto, debido a que está orientada con el principal propósito de mejorar los diversos sentidos del paciente, permitiendo así, efectuar el desarrollo de dos aplicaciones, una web y otra móvil, para poder complementar dicha aplicación; fundamentando su uso y esquematizando para poder ser un proyecto con gran potencial de escalabilidad, con el principal objetivo de mejorar la calidad de vida del usuario paciente, sea mejorando sus sentidos, e incluyéndolo a la sociedad por medio de estas herramientas.

De tal manera que, toda la información que se recopile por medio de las aplicaciones, almacenarla y centralizarla, para que su uso sea para beneficio de toda la comunidad que rodea a dicha audiencia, sean familiares, doctores o tratantes.

La aplicación web, fue desarrollada por medio del "*"Framework"*" "*"ASP.NET"*", que está ambientando en el lenguaje de programación "*C#",* a diferencia de las aplicaciones móviles, las mismas que fueron desarrolladas dentro del "*"Framework"*" "*Android Studio*", ambientadas en el lenguaje de programación "*Java*".

ABSTRACT

This project consists of the implementation of technological tools, developed by the author, to be able to provide a therapy, exclusively to a senior audience (60 years and older). Thus, fundamentally, an application developed by the author was implemented, which was a final project for a subject of the carriage, called "*Mobile Applications*", the same that generated a great impact, because it is oriented with the main purpose of improving the patient's various senses, thus allowing the development of two applications, one website and one mobile, in order to complement that application; basing its use and outlining to be a project with great potential for scalability, with the main objective of improving the quality of life of the patient user, either by improving their senses, and including it to society through these tools.

So, all information collected through the applications, store it and centralize it, so that its use is for the benefit of the entire community surrounding that audience, whether family members, doctors or traffickers.

The web application was developed by means of the "*“Framework”*" "*“ASP.NET”*", which is set in the programming language "*C*", unlike the mobile applications, the same ones that were developed within the "*“Framework”*" "*Android Studio*", set in the "*Java*" programming language.

ÍNDICE

1. CAPITULO I. INTRODUCCIÓN 1

1.1. Alcance 1

1.2. Antecedentes 2

1.3. Justificación 2

1.4. Objetivos 3

1.4.1. General 3

1.4.2. Específicos 3

2. CAPITULO II. MARCO TEÓRICO 4

2.1. Lenguajes de Programación. 4

2.1.1. C# 4

2.1.2. LINQ 4

2.1.3. Java 4

2.1.4. JSON 5

2.1.5. Lenguaje SQL 6

2.2. "*Framework*" 6

2.2.1. .NET 6

2.2.2. "*ASP.NET*" 6

2.2.3. "*ASP.NET*" MVC 7

2.2.4. "*ASP.NET*" RAZOR 7

2.2.5. "*Code First Entity Framework*" 8

2.2.6. Android Studio. 8

2.2.7. Laravel. 8

2.3. Web servicies 9

2.3.1. SOAP 9
2.3.2. REST 9
2.4. Herramientas 10
2.4.1. Navicat. 10
2.4.2. SQL Server 10
2.4.3. MySQL 10
2.4.4. Power BI 11
3. CAPITULO III. ANÁLISIS Y DISEÑO. 11
3.1. Definición de funcionalidad App "*RukusGame*" 11
3.2. Desarrollo 12
3.3. Funcionamiento. 12
3.3.1. Login 12
3.3.2. Menú Principal. 13
3.3.3. Doble o Nada 14
3.3.4. Simón dice 15
3.4. Estructuración de lógica de funcionamiento. 16
3.4.1. Estructuración de lógica de la aplicación web. 17
3.4.2. Estructuración lógica aplicación móvil "*ControlTarea*" 17
3.4.3. Estructuración lógica aplicación móvil "RukusGame" 18
3.5. Migración de servidores de base de datos. 18
3.6. Elaboración de la base de datos. 20
3.6.1. Tabla Actividad 20
3.6.2. Tabla "*Agendamientotarea*" 21
3.6.3. Tabla "*Agendamientotarea_user*" 21
3.6.4. Tabla "*FichaMedica*" 21

3.6.5. Tabla “*FichaMedica_user*” 21
3.6.6. Tabla “*ListaTratentes*”..... 22
3.6.7. Tabla “*Roles*”. 22
4. CAPITULO IV. Desarrollo de las aplicaciones. 22
4.1. Desarrollo de app web sobre “ASP.Net” 22
4.1.1. “*AdminLTE*” sobre visual Studio 22
4.1.2. Modelos 23
4.1.3. Creación de Contexto de base de datos. 24
4.1.4. Controlador 25
4.1.5. Vistas 27
4.1.6. Creación Controlador – Vistas “ASP.NET” MVC5 27
4.1.7. Desarrollo de servicios 29
4.2. Desarrollo de la app móvil 31
4.2.1. Librerías y consumo de servicios Web 31
4.2.2. Diseño Visual 32
4.3. Adaptación aplicación “*RukusGame”* 34
4.4. Desarrollo de análisis de información 34
4.4.1. Desarrollo de informe de estadísticas de los juegos en general ... 35
4.5. Infraestructura de la aplicación 35
5. CAPITULO V. Resultados. 38
5.1. Funcionamiento App Web “*PlataformaMED*” 38
5.2. Funcionamiento App Móvil “*ControlPaciente*” 41
5.3. Funcionamiento App Móvil *“RukusGame”* 42
6. Conclusiones y Recomendaciones 43

6.1. Conclusiones 43
6.2. Recomendaciones 44
REFERENCIAS 46
ANEXOS 48

ÍNDICE DE FIGURAS

Figura 1. Log in “*RukusGame*” 13

Figura 2. Menú principal “*RukusGame*” 13

Figura 3. Doble o Nada, selección de dificultad. 14

Figura 4. Pantalla del juego Simón dice. 15

Figura 5. Estructura Lógica de la aplicación Web. 17

Figura 6. Estructura Lógica App "*ControlTarea*" 17

Figura 7. Estructura lógica App "RukusGame" 18

Figura 8. Diagrama entidad relación base datos “gamesetdb” 19

Figura 9. Configuración de la cadena de conexión de la app web 25

Figura 11. Creación de controladores Visual Studio 28

Figura 12. Definición de agregación de controlador MVC 28

Figura 13. Menú Ver Itinerario. 32

Figura 14. Menú Registrar Fin Tarea app "ControlUsuario" 33

Figura 15. Control Actividad app ControlPaciente 34

Figura 16. Servicios Disponibles Microsoft Azure 36

Figura 17. Creación de nuevo grupo de recursos 37

Figura 18. Selección de plan y finalización de configuraciones 37

Figura 19. SQL Databases 37

Figura 20. Creación de base de datos, con sus respectivos atributos. 38

Figura 21. Login app Web “PlataformaMED” 39

Figura 22. CRUD Usuarios. 39

Figura 23. CRUD Listado tratantes. 40

Figura 24. Presentación de Reportería 41

Figura 25. Diagrama Entidad relación de base de datos “*PlataformaMed_db*” (en base al modelo anterior). 14

Figura 26 Lista Usuarios app ControlPaciente. 15

Figura 27. Lista de tareas app "ControlPaciente" 16

Figura 28. Notificación de Juego. 16

Figura 29. Reporte de estadísticas Simón dice, tanto por filtrado de agendamiento como vista general. 17

Figura 30. Recta de sesiones en función al tiempo. 18

Figura 31. Calendario de disponibilidad pacientes. 19

1. CAPITULO I. INTRODUCCIÓN

1.1. Alcance

El presente trabajo pretende desarrollar una aplicación web y móvil para la plataforma Android, la cual está enfocada en dos diferentes módulos de contenido, dentro de los cuales está completamente construido para poder reforzar el uso de la memoria, coordinación mano-ojo, concentración y motricidad. La aplicación consta de dos juegos iniciales:

1. “Doble o nada”, el cual consiste en encontrar todas las figuras iguales en diferentes modalidades de dificultad. El resultado de esto es remitido a un servidor, que permitirá alojar información relevante como tiempo de resolución, número errores, entre otros.

2. “Simón dice”, el mismo que consiste en apretar adecuadamente un color mostrado en la mitad de la pantalla. El color que se muestra dentro de la pantalla es generado aleatoriamente al iniciar el juego, lo cual hace que el adulto mayor use concentración, reflejos, motricidad, coordinación mano – ojo. Una vez culminado la sesión se envía la información relevante al servidor, tal como número de aciertos, número de errores, número de colores no contestados, tiempo de reacción entre cada intervalo de tiempo a responder.

La aplicación web está orientada en poder analizar toda la información recibida por los módulos de la aplicación móvil y esta enfocadas en poder mostrar un entorno más amigable. Dicha información elaboras una herramienta de análisis para el médico tratante, facilitando así la toma de decisiones y tratamiento. Toda la información será almacenada en la nube, para una accesibilidad global de todo usuario de tipo tratante, médico, evaluador, entre otros.

La aplicación “Médico – Persona Tratante”, es una aplicación móvil diseñada directamente en poder administrar correctamente las actividades del paciente, el mismo que usa los módulos de terapia dentro de la plataforma, para lo cual se permita administrar mediante vía web, o dentro de la misma; a la par, que el

usuario “Persona tratante”, tendrá ciertas limitaciones consideradas en la administración de actividades del paciente

1.2. Antecedentes

Actualmente, la comunidad conformada por personas que tengan una edad que supera los 60 años, en su mayoría de los casos se encuentra parcialmente (por no decir totalmente) aislada de la sociedad, ya que en la actualidad se rige por el manejo de diversos dispositivos tecnológicos y el correcto uso del internet de las cosas. Lo cual hace esto que el adulto mayor no tenga contacto general con la sociedad en cuestión, limitando sus medios de comunicación, capacitación y alcance de información respecto a los diferentes medios, y eternos que pueden aportar estos dispositivos electrónicos.

Actualmente, en nuestro país no existe una investigación enfocada en la tecnología y el adulto mayor la misma que permita reconocer los beneficios que tiene en este público, sobre todo en el tratamiento de distintos déficits médicos que se desarrollan a medida que la edad avanza. Para todo esto, el desarrollo de diversas soluciones tecnológicas será fundamental, ya que permitirá resolver varias necesidades de este grupo social, el mismo que abarca el 6.5% y que será el 18% para el 2050, lo q indica que se encuentra en incremento constante

1.3. Justificación

Gracias a la necesidad de poder implementar diversas herramientas tecnológicas que faciliten la evaluación y otorgar una mejoría a los diversos sentidos humanos, que posee el adulto mayor, tales como: motricidad, coordinación mano – ojo, memoria, concentración; a la par de poder mostrar los presentes resultados dentro de una plataforma o aplicación web, se desarrolla el presente proyecto, el mismo que permite el análisis de datos arrojados en tiempo real, que permiten mostrar al médico o persona a cargo del usuario de la aplicación información que permita emitir diagnósticos.

Por medio de la tecnología el adulto mayor puede integrarse de mejor manera en la sociedad, permitiéndole adquirir una nueva herramienta que le permita desenvolverse de mejor manera en su diario vivir; es decir, mejorando radicalmente su calidad de vida

1.4. Objetivos

1.4.1. General

Desarrollar una aplicación web y móvil (para la plataforma Android), la misma que se convierta en una herramienta que permita evaluar a través de diversas métricas el desempeño del usuario, que en este caso es un adulto mayor (edad comprendida entre los 60 años en adelante).

1.4.2. Específicos

Implementar y optimizar el código de la aplicación móvil, eliminando errores y posibles fallas presentes dentro de la misma.

Diseño de la aplicación web, que pueda cumplir con las necesidades referentes al tipo de usuario “tratante”, para una mejor administración de la información y de los usuarios en cuestión.

Diseño de una aplicación móvil, para la plataforma Android, la cual sirva como manejo de las actividades a cumplir por parte del paciente, la cual tendrá dos tipos de administración, una como “Médico”, y otra como “Persona tratante”, mismas se diferenciarán sobre el poder de administrativo sobre las actividades del paciente.

Presentar información relevante, a través de la plataforma o aplicación web, acerca de todos los datos recopilados referente de cada usuario a cargo del usuario tratante.

2. CAPITULO II. MARCO TEÓRICO

2.1. Lenguajes de Programación.

2.1.1. C#

"*C#*" es un lenguaje orientado a objetos que permite a los diversos desarrolladores elaborar una gran variedad de aplicaciones, tanto seguras como robustas, las mismas que trabajan sobre ".Net "Framework"". (Microsoft Documents, 2019)

2.1.2. LINQ

"*Language-Integrated Query*" (LINQ) es el nombre de un conjunto de tecnologías, dedicadas a la integración de capacidades de consulta hacia una fuente de datos, sea un servidor de tipo SQL, o incluso información perteneciente de archivos con extensión "*.xml*" (Microsoft Documents, 2019)

2.1.3. Java

"*Java*" es un lenguaje de programación, que permite definir diversas instrucciones, usando comandos (basados en el idioma inglés) para la computadora, en lugar que tener que enviar o escribir códigos numéricos.

Este lenguaje es altamente identificado como "lenguaje de alto nivel", debido a que el usuario, puede leerlo, interpretarlo, y escribirlo muy fácilmente. (Leahy, 2019).

A la par, este lenguaje es conocido como un lenguaje orientando a objetos, lanzado al mercado oficialmente en enero de 1996, por la empresa "*Sun Microsystems*". Este lenguaje de programación puede ser adquirido muy fácilmente desde internet y de forma libre o gratuita. Vale recalcar que cualquier versión contiene elementos básicos muy utilizados para la creación de cualquier aplicación existente para este, la misma que es muy amplia.

Por medio del presente lenguaje, se pueden realizar dos tipos distintos de aplicaciones.

- **Aplicaciones**
  Son programas que pueden ser implementados en cualquier equipo, haciendo uso de un intérprete del presente código, o a su vez una complicación de la solución para poder ser ejecutada.
- **Applets**
  Son programas que pueden integrarse fácilmente en páginas web o apps webs, caracterizadas por ser ejecutadas por el lado del usuario o cliente.

Este lenguaje de programación presenta diversas características, las cuales son:

- Es un leguaje orientado a objetos
- Es un lenguaje de tipo concurrente
- Se caracteriza por su amplia integración de objetos, de tipo clase, para la creación GUIS, orientadas para el usuario final.
- Es un lenguaje simple, con amplia integración y aplicación de seguridad, y muy robusto.
- Su sintaxis precede, o tiene rasgos de similitud con el lenguaje de programación "*C*++", con la gran ventaja que no presenta ningún problema o "*bug*" referente a "*C*++".
- La gran mayoría de las soluciones (programas) dentro de la red implementan el presente lenguaje. (Leahy, 2019)

2.1.4. JSON

"*Java Script Object Notation*" es un formato de información, con el objetivo de intercambiar datos entre diversas plataformas. El presente formato es caracterizado por ser sumamente ligero, a su vez, que para el usuario implementador es muy fácil leer, interpretar e incluso escribir el mismo.

El formato de "*JSON*", es caracterizado por ser un lenguaje libre e independiente, pero utiliza diversas terminologías que son frecuente implementadas por los programadores, con conocimiento cualquier lenguaje de programación. (JSON.org , 2019)

2.1.5. Lenguaje SQL

El Lenguaje SQL o "*Structured Query Language*", en español denominado, lenguaje estructurado de consultas, diseñado para la administración y localización de información dentro de los diversos sistemas de gestión de base de datos (DBMS).

Este lenguaje de programación es de tipo declarativo; es decir, sólo se debe indicar mediante instrucciones, la definición estructura necesaria que forma la base de datos, o la consulta de información requerida. (Escofet 2019)

2.2. "*Framework*"

2.2.1. .NET

"*.NET" "Framework"*" es un entorno de desarrollo, provisto por "*Windows*", que brinda una gran variedad de servicios a las aplicaciones que se ejecutan dentro del mismo.

Este entorno proporciona una gran variedad de bibliotecas de código, para que el desarrollador pueda implementar dentro de sus soluciones de manera gratuita y de fácil accesibilidad, a la par, de poder agregar alguna biblioteca requerida por medio de un amplio catálogo de bibliotecas alojadas en la nube. (Microsoft Documents, 2019)

2.2.2. "*ASP.NET*"

"ASP.NET" Core es un "*"Framework" multiplataforma*"; es decir, un marco de trabajo que permite desarrollar diversas aplicaciones modernas, basadas para específicamente para la nube y conectadas al internet., considerado óptimo y, sobre todo, de tipo "código abierto".

Con "ASP.NET" Core, posee la posibilidad de:

- Desarrollo aplicaciones, servicios web, aplicaciones "*IoT*" y "*backends*" móviles.

- Posibilidad de desarrollo en cualquier sistema operativo de distribuciones Windows, Linux y Mac.
- Implementación en la nube o "*On premises*".
- Se puede implementar en .NET Core o .NET "Framework". (Microsoft Documents, 2019)

2.2.3. "*ASP.NET*" MVC

El patrón de tipo arquitectónico "*Model - View - Control*" (MVC), segmenta una aplicación en tres grupos principales de componentes:

- Modelos
- Vistas
- Controladores.

Este patrón ayuda a lograr la separación de cualquier error enfocada a cada uno de estos componentes, y la lógica que está atada detrás de cada uno. Usando este patrón, las solicitudes de los usuarios se enrutan a un Controlador que es responsable de trabajar con el Modelo para realizar las solicitudes propuestas por los usuarios y / o recuperar los resultados de las consultas. El Controlador elige la Vista para mostrar al usuario y le proporciona los datos del Modelo que necesita.

Esta definición o distribución de responsabilidades facilita el escalamiento de la aplicación en términos de complejidad, ya que, para el usuario, al momento de programar facilita para el desarrollador codificar, depurar y probar (modelo, vista o controlador) un solo flujo de trabajo para obtener el resultado deseado.

Es más difícil actualizar, probar y depurar el código que tiene dependencias distribuidas en dos o más de estas tres áreas, debido que se debe localizar el problema, error latente en varias instancias para identificar el código. (Microsoft Documents, 2019)

2.2.4. "*ASP.NET*" RAZOR

"*Razor Pages*" es un nuevo atributo de "ASP.NET" Core MVC que hace que la codificación de escenarios enfocados en páginas sea más fácil y productiva.

"*Razor"* de *"ASP.Net"*, "*MVC*" es un considerado un motor de visualización, el cual por medio de sintaxis permite escribir el código del lado del servidor, explícitamente en la vista. Ayuda combinar código y "*html*" de manera fluida.

Vale recalcar, que la presente herramienta no es considerada como un lenguaje de programación, pero si el usuario tiene conocimientos básicos en "*C #*", "*Vb.Net*" y "*HTML*", puede interpretar y escribir fácilmente código "*Razor*".

Por medio de este motor, puede generar páginas web de manera fácil y sencilla en función a modelos establecidos por medio del intérprete del mismo, facilitando el desarrollo visual para el implementador. (Microsoft Documents, 2019)

2.2.5. "*Code First Entity Framework*"

"*Code-First*" es un estilo de programación enfocado para uso exclusivo de la aplicación a desarrollar, para lo cual en su principio de desarrollo consiste en crear clases para definir entidades de dominio, en lugar de diseñar una base de datos en primera instancia, para que luego se migre automáticamente a un servidor de BD (base de datos), permitiendo así generar una instancia completamente nueva y un ambiente de desarrollo factible para la aplicación en desarrollo. (Microsoft Documents, 2019)

2.2.6. Android Studio.

Android Studio, es "*IDE*" (entorno de desarrollo integrado), reconocido de manera oficial para el desarrollo de aplicaciones para sistemas operativos de tipo "*Android*" distribuidos por Google, dentro del cual alberga un sinfín de herramientas de diseño orientado para el "*back-end*" y el "*front-end*" de la aplicación a desarrollar, incluso provee de una herramienta exclusiva para poder virtualizar un dispositivo móvil de distribución "*Android*" para las respectivas pruebas pertinentes. (Android Developers, 2019).

2.2.7. Laravel.

Laravel es un "Framework" de desarrollo para aplicaciones, ambientadas para el lenguaje "*PHP*". Debido a la facilidad de integración de diversas funcionalidades, y fácil manejo del lenguaje "*PHP*", hace que este "*"Framework"*" sea uno de los

más implementados para el manejo y desarrollo de este tipo de aplicaciones. (Rees, Dayle, and Antonio Laguna).

2.3. Web servicies

Un "*Web service*" (servicio web), es una herramienta tecnológica, que implementa un conjunto de estándares y protocolos para todo intercambio de información entre aplicaciones, sin importar en que lenguaje este desarrollado; siempre y cuando la misma pueda interpretar esta herramienta. Dentro de estas herramientas tecnologías, se destacan dos servicios, SOAP y REST. (Alonso, Gustavo)

2.3.1. SOAP

Un servicio web "*SOAP*" tiene una interfaz que se describe en un formato procesable por máquina que se denomina "*documento de lenguaje de definición de servicios web"* (WSDL). Un servicio web "*SOAP*" se describe utilizando una noción formal y estándar de "*XML*" que proporciona todos los detalles necesarios para interactuar con el servicio, incluidos los formatos de mensajes, los protocolos de transporte y la ubicación. Las herramientas se pueden usar para procesar el "*WSDL*" y producir programas cliente capaces de comunicarse con el servicio utilizando el protocolo "*SOAP*" basado en "*XML*". "*SOAP*" puede ser un protocolo de comunicación detallado, pero tiene la ventaja de la extensibilidad; existen más especificaciones para admitir las calidades de servicio de la empresa, como el soporte distribuido en dos fases y las opciones de seguridad sofisticadas. (IBM 2018)

2.3.2. REST

"*Representational State Transfer*" (REST) es un servicio de estilo arquitectónico que, especifica restricciones, como la interfaz uniforme, que, si se aplica a un servicio web, dicha arquitectura cambio de manera crucial la forma en la que se desarrolla el servicio web, esquematizando, y organizando de mejor manera la forma en la que el servicio se comparta, de tal forma que, se los atribuye cómo

se comporta el servidor al momento de adquirir una petición por parte del usuario final. (Oracle, 2019).

2.4. Herramientas

2.4.1. Navicat.

"*Navicat*" es una herramienta de desarrollo y gestión de base de datos, que permite conectar de manera simultánea a diversos servidores de base de datos, según como especifique el usuario, dentro de lo servidores existentes se tienen:

- MySql
- MariaDB
- MongoDB
- SQLServer
- Oracle
- Posgrade
- SQLite

Vale recalcar, que esta herramienta puede conectarse incluso con servidores en la nube o remotos, siempre y cuando tenca acceso. (Navicat Premium, 2019).

2.4.2. SQL Server

Microsoft SQL Server es un servidor dedicado a la administración y gestión de bases de datos desarrollado por Microsoft; con la principal funcionalidad de almacenar y recuperar datos según lo solicite el usuario (sea internamente, o por medio de otras aplicaciones), el mismo que permite centralizar la información para todo un sinfín de software según lo necesite. (Microsoft Documents, 2019)

2.4.3. MySQL

MySQL es un servidor de bases de datos SQL caracterizado por ser rápido, multiproceso, multiusuario, robusto y, sobre todo, por ser un software de libre acceso.

El presente software es uno de los más implementados en el mercado, debido a su costo y el nivel de accesibilidad que posee el mismo. (MySQL Developers, 2019).

2.4.4. Power BI

Es un aplicativo integro para el análisis de tipo empresarial; permite al usuario visualizar datos definidos por el mismo, y a su vez, compartir información dentro de la organización a la cual pertenece, o simplemente publicarlos en un sitio web o aplicación.

Permite conectarse a diversas fuentes de datos, y con una facilidad de poder visualizar de manera "*live*" o "en vivo" para cualquier usuario. (PowerBI, 2019).

3. CAPITULO III. ANÁLISIS Y DISEÑO.

3.1. Definición de funcionalidad App "*RukusGame*"

La app "*RukusGame*" fue un proyecto desarrollado como tema de investigación de proyecto final dentro de la materia de "*Aplicaciones móviles*" en la carrera de Ingeniería Electrónica y Redes de la Universidad de las Américas, por ende, su principal objetivo fue desarrollar dos juegos.

El primero denominado "*Simón Dice*", y otro denominado "*Doble o nada*", con el propósito de que estos juegos permitan generar terapias orientadas principalmente para el adulto mayor, con la finalidad de poder tratar, mejorar y evaluar por medio de la recepción de información del juego, como se encuentra la motricidad, concentración, lucidez, visión, coordinación mano – ojo del paciente.

La presente aplicación consume servicios de tipo "*REST*", los cuales permite capturar dentro de una aplicación que administra la información recopilada por la misma app, por medio de formato "*JSON*", y la almacena dentro del servidor de base de datos.

Finalmente, la aplicación web y la aplicación móvil consisten en están orientadas para el usuario de tipo médico, o usuario de tipo tratante (con limitación a

administrar inicio y fin de tarea) para poder integrar, facilitar y complementar el trabajo desarrollado a lo largo de esta materia, para poder realizar un proyecto integrador, innovador y revolucionario para los involucrados.

3.2. Desarrollo

El funcionamiento principal de la aplicación "*RukusGame*", está basada su desarrollado en lenguaje de programación Java, por medio de la plataforma Android Studio, para la elaboración de aplicaciones móviles en el sistema operativo "*Android Studio*", distribuido por Google,

Dicha aplicación está definida con una estructuración de modelos, clases de tipo controladores, y diseños de interfaz gráfica netamente desarrollado por medio de etiquetas XML, tal como lo plantea la plataforma "*Android Studio*".

Para lo cual se vio necesario una librería denominada "*java.net.HttpURLConnection*", para poder realizar el respectivo consumo de servicios de tipo REST, para poder almacenar la información recopilada por ambos juegos.

3.3. Funcionamiento.

3.3.1. Login

Dentro de esta forma, el usuario puede autenticarse, para hacer uso de las aplicaciones, y con la peculiaridad de un botón de tipo "*configuración*", el mismo que permite definir el enlace donde se encuentra alojado los servicios web a consumir. Tal como se muestra en la Figura 1.

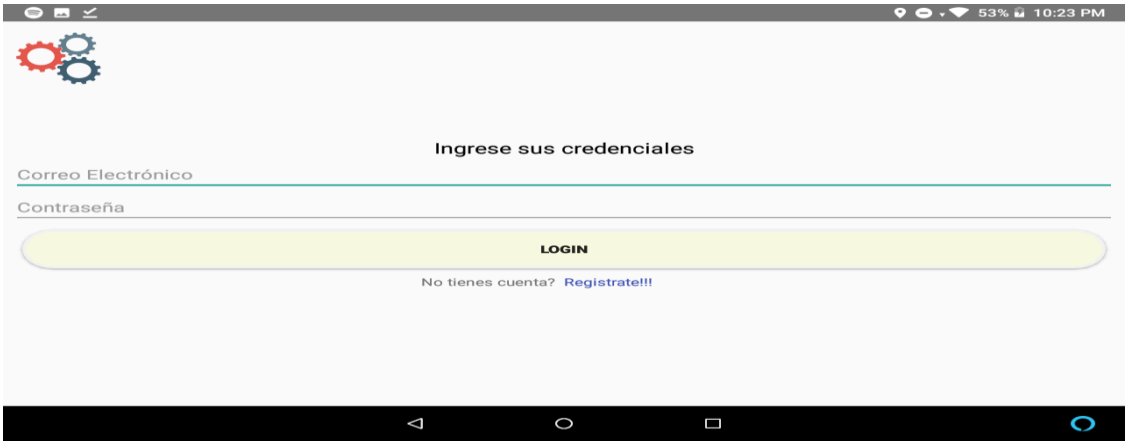
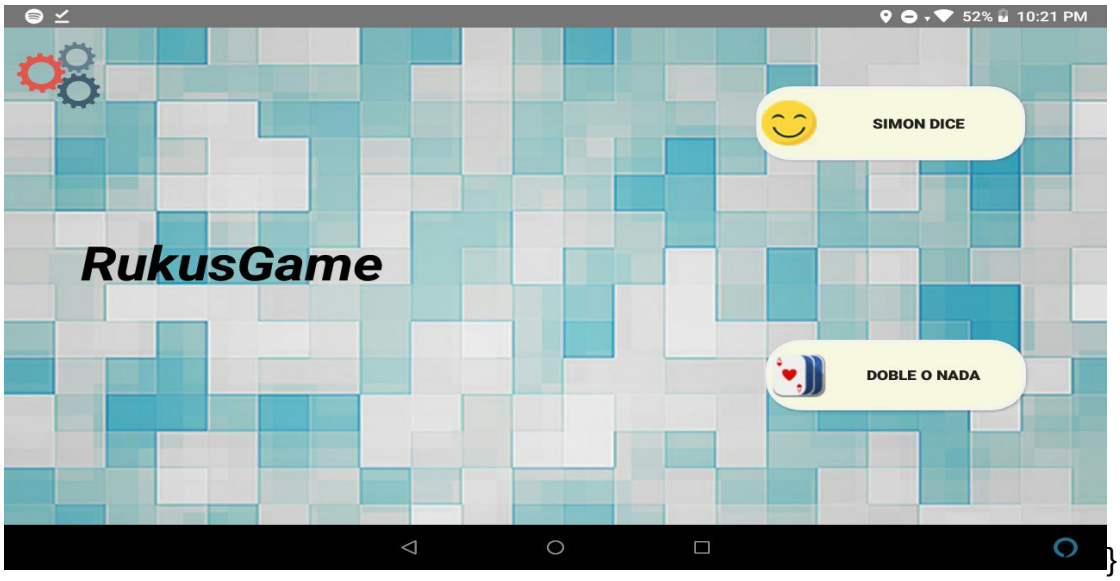


Figura 1. Log in "*RukusGame*"

Al presionar el botón de configuración (en forma de engranaje), como se muestra en la figura 3, se desplegará un menú, dentro del cual nos permite ingresar la dirección de tipo URL, para poder referenciar adecuadamente el consumo de los servicios web, necesario para el funcionamiento de la aplicación.

3.3.2. Menú Principal.

Dentro de la presente pantalla cumple con la funcionalidad y poder seleccionar el juego que se desea aplicar, se tienen dos opciones con botones visuales personalizados, para presentar una interfaz más amigable para el usuario. Como se muestra en la Figura 2.



}

Figura 2. Menú principal "*RukusGame*"

3.3.3. Doble o Nada

Es un juego diseñado para ejercitar la memoria del jugador. El juego se estructura en 3 niveles distintos (fácil, intermedio, y difícil), para lo cual, está establecido como objetivo principal, encontrar todos los pares iguales dentro de una matriz de cartas, distribuidas en número de pares, tal como se muestra en la Figura 3.

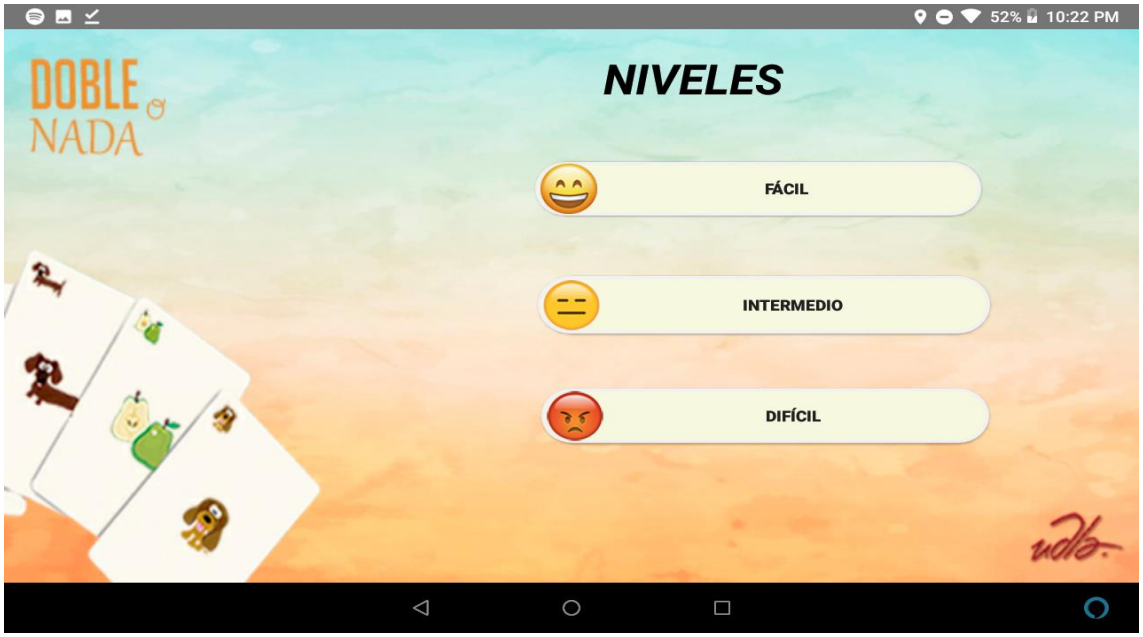


Figura 3. Doble o Nada, selección de dificultad.

- **Fácil**
  La matriz está establecida por 3 filas y 4 columnas, el cual el usuario debe encontrar los 6 pares establecidos para completar el juego.
- **Intermedio**
  La matriz está establecida por 4 filas y 4 columnas, el cual el usuario debe encontrar los 8 pares establecidos para completar el juego.
- **Difícil**
  La matriz está establecida por 4 filas y 5 columnas, el cual el usuario debe encontrar los 10pares establecidos para completar el juego.

Características especiales

- El juego se caracteriza por una peculiaridad incluida dentro de este, la cual consiste en voltear la primera carta seleccionada después de cierta cantidad de tiempo, dependiendo la dificultad, la carta se oculta más rápido; en dificultad fácil, la tarjeta gira a los 8 segundos, en dificultad intermedio, la tarjeta se voltea a los 6 segundos, y en difícil la tarjeta se voltea cada 3 segundos. Esta característica permite evaluar la concentración y lucidez, para incentivar al adulto mayor a desarrollar de mejor manera la terapia.

3.3.4. Simón dice

El juego permite mejorar la motricidad, coordinación mano - ojo y concentración. Para esto la interfaz del juego muestra cuatro alternativas de colores de las cuales el jugador debe seleccionar la respuesta solicitada por la aplicación, por medio de un visor, en este caso colores. Como se puede apreciar en la Figura 4.

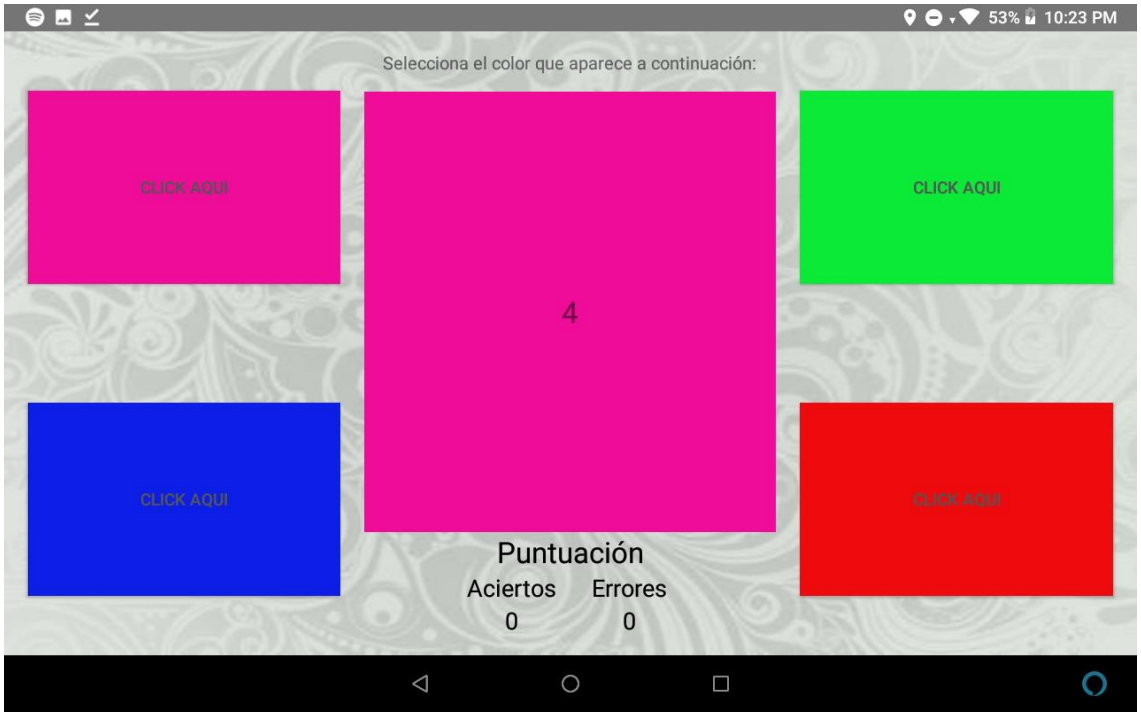


Figura 4. Pantalla del juego Simón dice.

La coordinación se evalúa en el transcurso del juego, para ello el mismo inicia con un tiempo de selección de cinco segundos en el primer intento y se reduce a un segundo para el intento número catorce.

"*Simon Dice*" se caracteriza en mostrar una serie de botones, un contenedor principal, el usuario deberá seleccionar adecuadamente el color o figura mostrada en el contenedor, antes de que el contador llegue a "*0*", marcará las estadísticas necesarias, tales como tiempo de reacción (tiempo en el cual el usuario respondió durante el contador), si fue un acierto, un error, o simplemente no respondió, para luego recopilar los datos por la sesión, y poder subirlos a la plataforma web.

3.4. Estructuración de lógica de funcionamiento.

El desarrollo de una aplicación web y móvil permite al usuario, de tipo administrador y médico

- Agregar nuevos usuarios de tipo tratante, o paciente
- Gestionar las sesiones de los pacientes
- Observar su progreso de forma sencilla.

Por otro lado, al usuario paciente, se le permite realizar su terapia fuera del consultorio y se envían mensajes de alerta en caso de olvidar su tratamiento gracias al funcionamiento conjunto de estas aplicaciones.

La aplicación web inicia con un inicio de sesión en el cual el medico puede planificar las sesiones de sus pacientes y además visualizar los datos obtenidos en el tiempo ya sean estos generales o por paciente.

La aplicación móvil enfocada netamente en la administración de las tareas agendadas para el usuario de tipo paciente, para lo cual simplemente permite llevar a detalle un registro de inicio de tarea, fin de tarea, con el objetivo de poder evaluar las diferencias de tiempo establecido con por el cual el paciente cumplido la tarea encomendada (puede ser inicio tarde de la actividad, fin temprano de la actividad, actividad prolongada), para que esta información pueda ser evaluada a detalle y ser implementada de la mejor manera, como el usuario administrador o médico crea necesario.

3.4.1. Estructuración de lógica de la aplicación web.

La aplicación web permite desempeñar diversos factores estratégicos para su funcionamiento, de tal forma que se detalla en la Figura 5.

Figura 5. Estructura Lógica de la aplicación Web.

3.4.2. Estructuración lógica aplicación móvil "*ControlTarea*"

La aplicación móvil permite desempeñar diversos factores estratégicos para su funcionamiento, de tal forma que se detalla en la Figura 6.

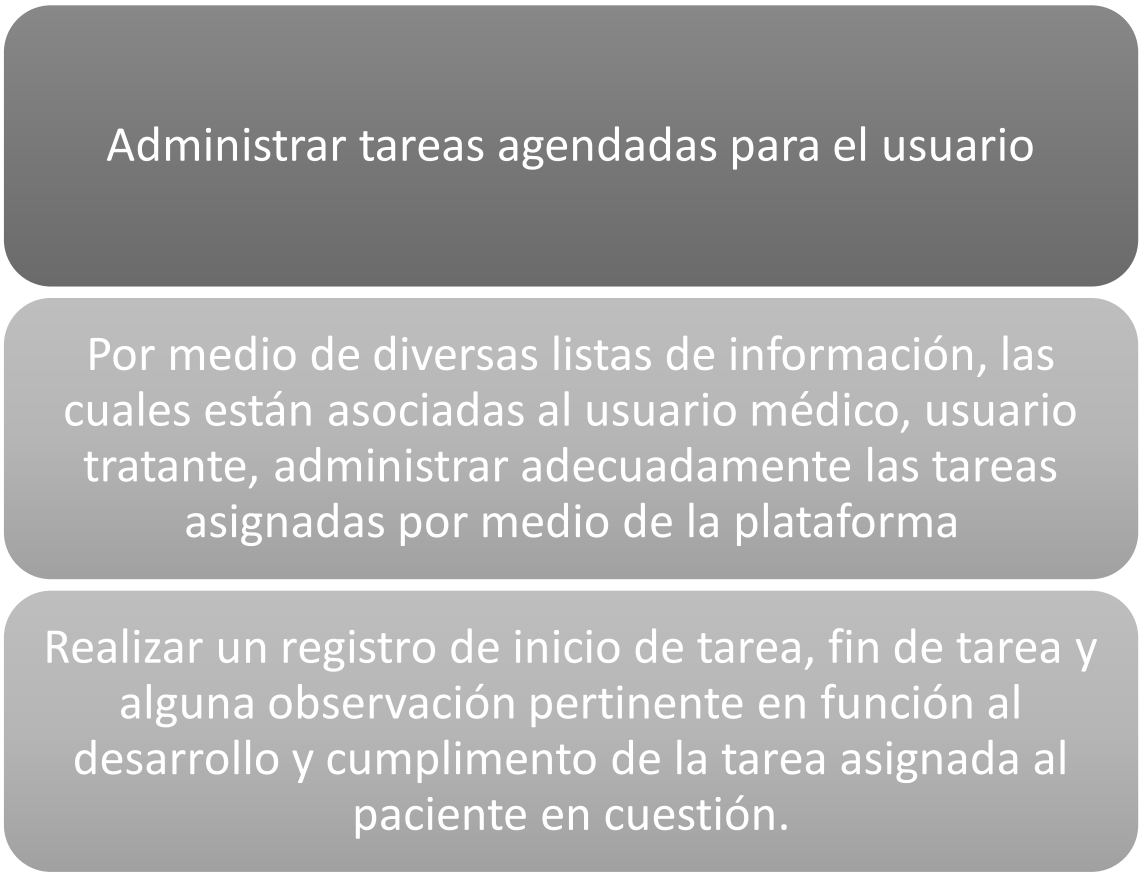


Figura 6. Estructura Lógica App "*ControlTarea*"

3.4.3. Estructuración lógica aplicación móvil “RukusGame”

La aplicación móvil “*Rukusgame*”, será adaptada al funcionamiento de la plataforma para un mejor control e integración de esta; de tal forma que, sus funciones están establecidas dentro de la Figura 7

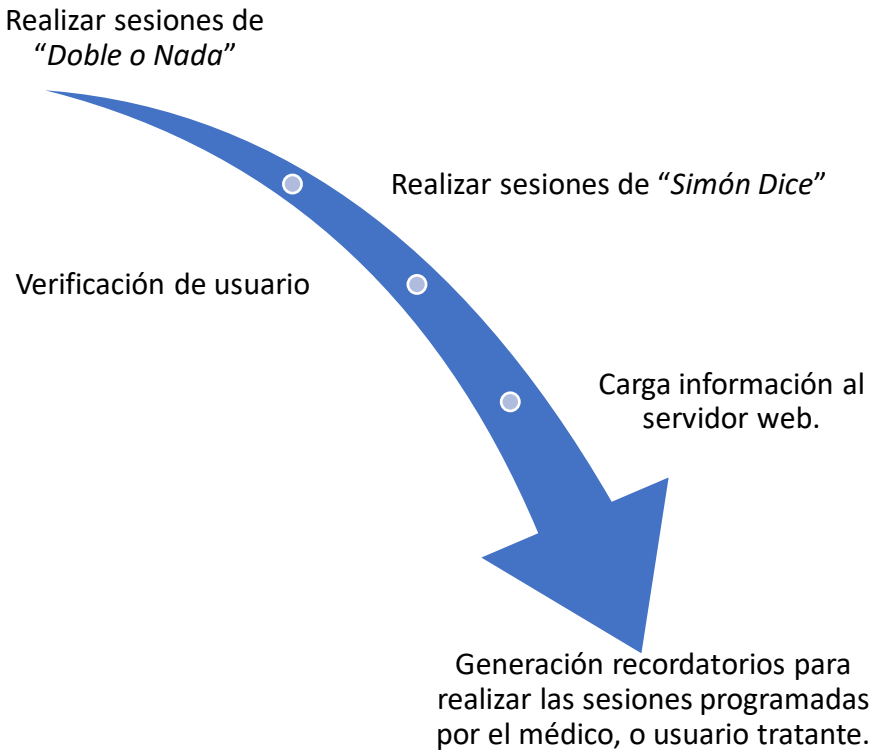


Figura 7. Estructura lógica App "*RukusGame*"

3.5. Migración de servidores de base de datos.

Mediante la herramienta “*Navicat*” se establece la conexión de base de datos entre MySQL y SQL Server. La herramienta permite exportar con facilidad la información y estructura de datos de los diversos servidores de bases de datos. Para el desarrollo del presente proyecto se implementó una migración desde un servidor de base de datos alojado en la nube, permitiendo ajustar, administrar y copiar la estructura necesaria para poder migrar dicha información a un servidor SQL Server, e integrar complemente a la plataforma desarrollada en “ASP.Net”, facilitando la integración de dichas herramientas debido a que son de un mismo fabricante.

Por ende, la estructura de base de datos está dada en la Figura 8

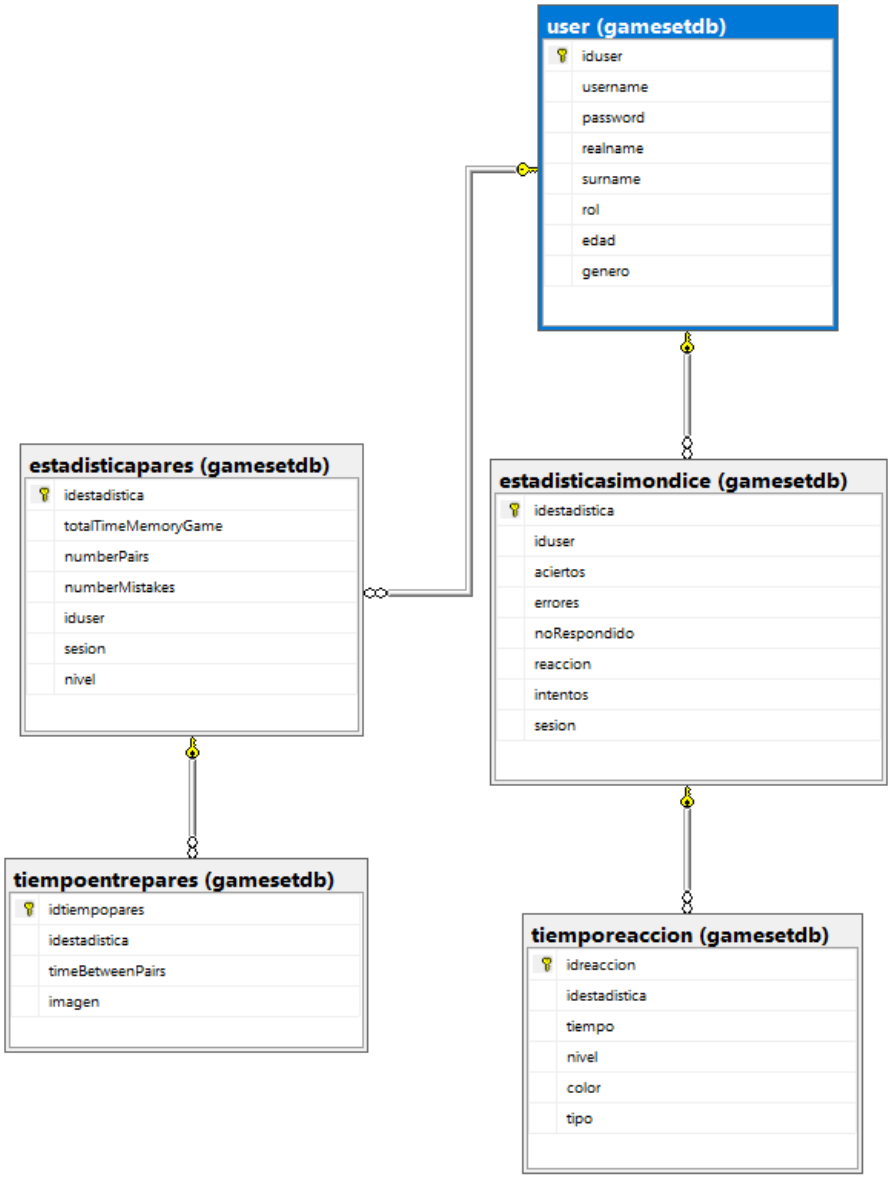


Figura 8. Diagrama entidad relación base datos “gamesetdb”

La presente base de datos fue desarrollada para el uso exclusivo de almacenamiento para el funcionamiento previo de la aplicación “*RukusGame*” para lo cual, cumple las necesidades básicas, dicha información era gestionada por una aplicación web desarrollada con el “*“Framework”*” “*Laravel*”, por código

PHP, para lo cual, solventaba las necesidades de almacenamiento y análisis de información.

3.6. Elaboración de la base de datos en función a las nuevas necesidades.

Con el fin de proveer al médico tratante las herramientas necesarias para la evaluación de sus pacientes se establecen diversas tablas y relaciones.

La base de datos desarrollada permite la administración de usuarios, agendamiento de sesiones y la generación de reportes.

Dicha base de datos fue desarrollada por medio de una técnica de programación denominada "*Code First*" la cual consiste en el desarrollo en conjunto de la solución en función de la aplicación; es decir, se desarrolla mediante código la estructura de base de datos en función al modelo implementado.

Vale recalcar que por medio de esta técnica de desarrollo se logró obtener el siguiente resultado mostrado en el Anexo 7, Figura 33.

Para lo cual el usuario médico tratante, por medio de las entidades:

- Actividad
- Agendamientotarea
- Agendamientotarea_user
- FichaMedica
- FichaMedica_user
- ListaTratentes
- Roles.

Para poder facilitar toda la funcionalidad necesaria para poder brindar una solución factible, específicamente a la necesidad de administración del usuario, y su respectivo itinerario, por medio de las aplicaciones desarrolladas para poder brindar una terapia al adulto mayor.

3.6.1. Tabla Actividad

Por medio de la presente entidad, está orientada almacenar toda la información relevante a tareas que puede realizar el usuario, para lo cual, el usuario de tipo

administrador podrá agregar paulatinamente más actividades en función de la integración de nuevas aplicaciones orientadas a brindar una terapia para cualquier paciente dado, indiferentemente de su edad.

Vale recalcar que el presente proyecto está orientado netamente para el tratamiento de pacientes de tercera edad.

3.6.2. Tabla "*Agendamientotarea*"

Por medio de la presente entidad, está orientada almacenar toda la información relevante al agendamiento de tarea, tal como fecha de inicio, fecha fin de la tarea, la actividad a desarrollar, observaciones de la tarea y el estado. Facilitando al médico monitorear la agenda programa por medio de la herramienta desarrollada

3.6.3. Tabla "*Agendamientotarea_user*"

Por medio de la presente entidad, se busca romper la relación muchos a muchos, presente entre la tabla "*agendamientotarea*" y la tabla "*user*", facilitando administrar de manera sensitiva y óptima las tareas agendas, permitiendo así asignar la tarea a un usuario de tipo paciente, usuario de tipo médico y en última instancia un usuario de tipo tratante.

3.6.4. Tabla "*FichaMedica*"

Por medio de la presente entidad, se platea almacenar toda la información, básica y relevante referente datos médicos del usuario de tipo paciente, para una referencia a los diversos usuarios de tipo médico o tratante.

3.6.5. Tabla "*FichaMedica_user*"

Por medio de la presente entidad, se busca romper la relación muchos a muchos, presente entre la tabla "*FichaMedica*" y la tabla "*user*", facilitando administrar de manera sensitiva y óptima la información médica necesaria, provista por cada tipo de usuario médico, facilitando así dar centralización de esta información vital para el mejor desempeño del paciente.

3.6.6. Tabla "*ListaTratentes*"

Por medio de la presente entidad, se platea almacenar una lista de usuarios de tipo paciente, asignados a usuarios de tipo tratante, para poder así, controlar de mejor manera el dominio que tiene cada usuario de tipo tratante, y a que usuarios se tiene permitido efectuar el tratamiento y control necesario de la tarea asignada por el médico.

3.6.7. Tabla "*Roles*".

Por medio de la presente entidad, se plantea almacenar todos los roles necesarios que servirán de referencia para la aplicación web, la misma que implementará para poder evaluar grados de permisos, para acceso a diversas funcionalidades provistas.

4. CAPITULO IV. Desarrollo de las aplicaciones.

4.1. Desarrollo de app web sobre "ASP.Net"

A través del "*"Framework"*" "ASP.Net" es completamente factible desarrollar la aplicación web, la misma que permitirá llevar acabo en control del usuario y el agendamiento respectivo de su agenda a tratar.

La implementación del presente "*"Framework"*" está orientado a la factibilidad, uso y desarrollo de manera intuitiva, y fácil para el desarrollador con conocimientos generales en lenguajes de programación C#, Java para el correcto uso de este, para lo cual, permite inclusive el uso de la aplicación sin importar la plataforma de destino.

4.1.1. "*AdminLTE*" sobre visual Studio

"*AdminLTE*", es una plantilla o "*template*", que da estilización y ciertas directrices de código (CSS y JS) para el uso dentro del desarrollo de esta, para lo cual facilita al usuario implementar todos los componentes que provee la presente plantilla en su forma lite (libre), de manera que, permite al usuario darle una estilización de aplicación móvil, a la app web.

Para poder implementar dicha plantilla, en el presente proyecto fue necesario, por medio de la herramienta "*Visual Studio Market Place*", adquirir el producto *"AdminLTE Template*" desarrollado por el usuario "*Connor O'Shea*", lo cual el licenciamiento y uso de la presente plantilla es de uso libre, sin fines de lucro.

Para poder adquirir la presente plantilla, es necesario ir a la página de "*Visual Studio Market Place"* o simplemente redireccionarse al siguiente enlace:

"*https://marketplace.visualstudio.com/items?itemName=c0shea.AdminLTETemplate"*

Una vez dentro del enlace, se procede a descargar el archivo de "*template*", para lo cual el presente archivo se comporta como un instalador, dentro del mismo, simplemente se debe seguir los pasos mostrados dentro del mismo, con el único requerimiento que debe estar cerrada la herramienta Visual Studio sobre el equipo en el cual se procederá a instalar.

Una vez concluida la instalación, al crear un nuevo proyecto dentro de la herramienta Visual Studio, se podrá apreciar al "*template*" con las configuraciones necesarias para poder implementar la plantilla "AdminLTE", con un proyecto de "ASP.Net".

Al momento de crear el nuevo proyecto, se puede apreciar la estructura MVC, sobre la exploración de soluciones.

4.1.2. Modelos

Una vez generado el proyecto, el primer paso a realizar, es la creación de clases de tipo "*.cs*", las mismas que serán las entidades para la creación de la base de datos por medio de "*Entity "Framework" Code First"*, para lo cual, se debe crear las clases, con atributo "*public*", a la par de identificar adecuadamente los atributos, y que tipo de variable almacenara en cada atributo a almacenar.

Ciertas clases estarán orientadas en el modelo de base de datos implementado dentro de la aplicación "*RukusGame*", a la par de la generación de las entidades necesarias para las funcionalidades específicas de la de aplicación web a desarrollar. De tal manera que se procede a tener los siguientes modelos.

Vale recalcar que es completamente necesario implementar la librería "*Data Annotation*", las mismas que nos permiten identificar el tipo de atributo, tal como:

- Si el atributo es de tipo clave primaria
- Si el atributo es de tipo clave foránea
- Si el atributo no puede ser de tipo "*null*"
- Si el atributo puede ser de tipo "*null*"

Entre otros.

A la par, para poder relacionar una tabla con otra por medio de claves foráneas se debe tomar en cuenta que, toda relación debe ser de uno a muchos, con la identificación que la clase que posea una clave foránea se debe especificar instanciando la clase de donde proviene, y la clase que atribuye la clave foránea, especificar que contendrá una colección de datos de la tabla que requiere la clave foránea.

La estructuración de dichas clases está especificada en el Anexo 6, del presente documento

4.1.3. Creación de Contexto de base de datos.

El contexto de la base de datos es una clase que permite a la aplicación definir los detalles necesarios para la creación de la base de datos, en función a los modelos desarrollados por el usuario

Dentro de dicho contexto, se define exactamente las tablas a desarrollar, en función a que modelo, en caso de que un modelo no esté registrado dentro de esta clase, simplemente la aplicación no lo desarrolla la tabla en cuestión.

Para la creación de la clase contexto, es necesario crear una nueva carpeta sobre la solución a tratar, en el caso de esta solución, la carpeta fue nombrada "*Context*", y dentro de este se crea la nueva clase.

Una vez definido los modelos dentro de la clase "*Dbcontexto*", se procede agregar la cadena de conexión al servidor de base de datos en donde se va a

alojar la base de datos a crear, para lo cual, dentro del archivo “*Web.Config*”, como se puede apreciar en la Figura 9.

Figura 9. Configuración de la cadena de conexión de la app web

Agregamos dentro del archivo la cadena de conexión para que la aplicación ya pueda funcionar adecuadamente con la creación de base de datos por medio del código desarrollado en los modelos.

4.1.4. Controlador

Dentro del desarrollo de los controladores, se implementó la herramienta de auto creación de controlador y modelos, existente en el entorno de “*ASP.MVC*” versión 5, la misma que permite a los usuarios crear “*CRUD’s*” (Create, Update, Delete) a partir de modelos previamente establecidos, para lo cual facilita al usuario implementar tanto controlador y vistas.

En el caso de controladores, crea automáticamente los controladores con las funciones necesarias para los CRUDS, de tal manera que, se tiene de la presente manera:

- **Index**

  El presente contralor permite recuperar la información del modelo para el cual fue desarrollado, con el objetivo específico de mostrar toda la información en la vista creada con el mismo nombre, para lo cual, dicha información es recuperada de la base de datos generada, por medio de consultas de tipo LINQ, facilitando el entendimiento y funcionalidad.
- **Create**

  Genera el controlador para poder realizar una inserción de datos dentro de la base de datos, para lo cual recibe la información necesaria, sea que requiera un parámetro extra para poder funcionar, y poder enviar la información necesaria para la creación de dicho registro, este tipo de información inclusive puede ser información relacionada a claves foráneas.

- **Create (On Post)**

  Este controlador esta específicamente diseñado para recibir todos los parámetros que se busca insertar, provenientes de la vista generada con el mismo nombre (create), para lo cual antes de realizar cualquier inserción
- **Details**

  El presente controlador recibe el identificador de la colección de datos a mostrar, para lo cual, enviará la información respectiva a la vista para poder ser visualizada
- **Update**

  El presente controlar, recupera el identificador de la colección de datos que se desea actualizar, para lo cual recupera el atributo de tipo identidad, y lo envía a la vista para que el usuario pueda actualizar los diversos atributos, a medida que lo requiera.
- **Update (On Post)**

  El presente controlador recupera el identificador de la colección de datos que se desea actualizar, a la par de todos los valores para que sean actualizados dentro de los atributos necesarios y confirma la actualización de datos.
- **Delete**

  El presente controlador recupera el identificador de la colección de datos que se desea borrar, para lo cual recupera el atributo de tipo identidad, y lo envía a la vista, para que el usuario pueda confirmar o no la eliminación.
- **Delete (On Post)**

  El presente controlador recupera el identificador de la colección de datos que se desea borrar, para lo cual recupera el atributo de tipo identidad, y lo envía a la vista, para que el usuario pueda confirmar o no la eliminación.

En algunos casos, para solventar temas de funcionalidad, se adaptó el esquema de controladores generados, para poder hacer uso de dicha información y administrar de mejor manera dicha información, para lo cual fue necesario la modificación de consultas, agregación de validaciones por código, y consultas necesarias para obtención de información.

Dentro de la presente solución vienen abreviados dos controladores con sus respectivas vistas, debido a la implementación del "*template AdminLTE*", los mismos que están diseñados para poder alojar la página principal (controlador home) y en caso de exista algún error dentro de la app (errores de tipo 404, 405, entre otros) sean desplegados al usuario para poder tomar las correcciones del caso.

4.1.5. Vistas

Dentro del desarrollo de los controladores, se implementó la herramienta de auto creación de controlador y modelos, existente en el entorno de "*ASP.MVC*" versión 5, la misma que permite a los usuarios crear "*CRUD's*" a partir de modelos previamente establecidos, para lo cual facilita al usuario implementar tanto controlador y vistas.

En el caso específico de vistas, se generó vistas en función a los controladores, las mismas que contienen elementos web generados por "*"ASP.NET" Razor*", que permiten manejar adecuadamente la información en función de la necesidad del proceso a realizar, ya sea insertar un nuevo registro, consultar un registro, mostrar todos los registros de los atributos, actualizar un registro dado, o eliminar un registro dado

4.1.6. Creación Controlador – Vistas "ASP.NET" MVC5

Dentro de la herramienta de Visual Studio, para poder generar los controladores y visitas en función a un modelo preestablecido, se debe ubicar sobre la carpeta controlador, dar clic derecho sobre esta, luego se seleccionará la opción "*Agregar…*", dentro de esta opción seleccionamos controlador; una vez realizado el proceso descrito con anterioridad mostrará en la Figura 10.

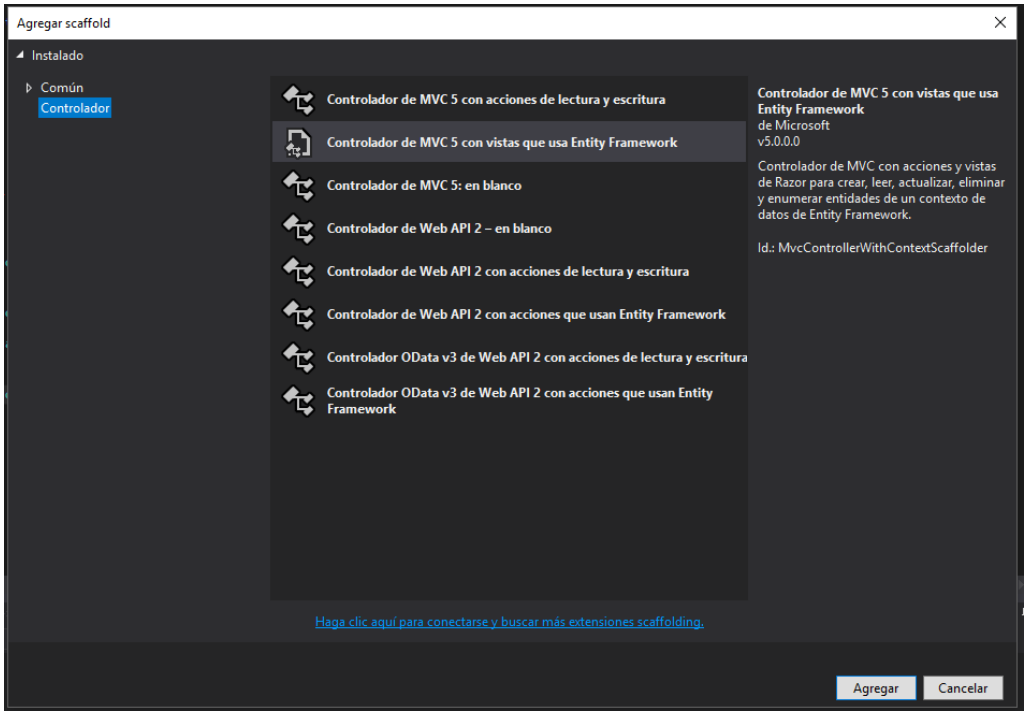


Figura 10. Creación de controladores Visual Studio

Dentro de la presente pantalla, es necesario seleccionar la opción "*Controlador de MVC5 con vistas que usan Entity "Framework"*", una vez seleccionada, damos clic sobre el botón agregar; una vez hecho esto, se desplegará la ventana dentro de la Figura 11.

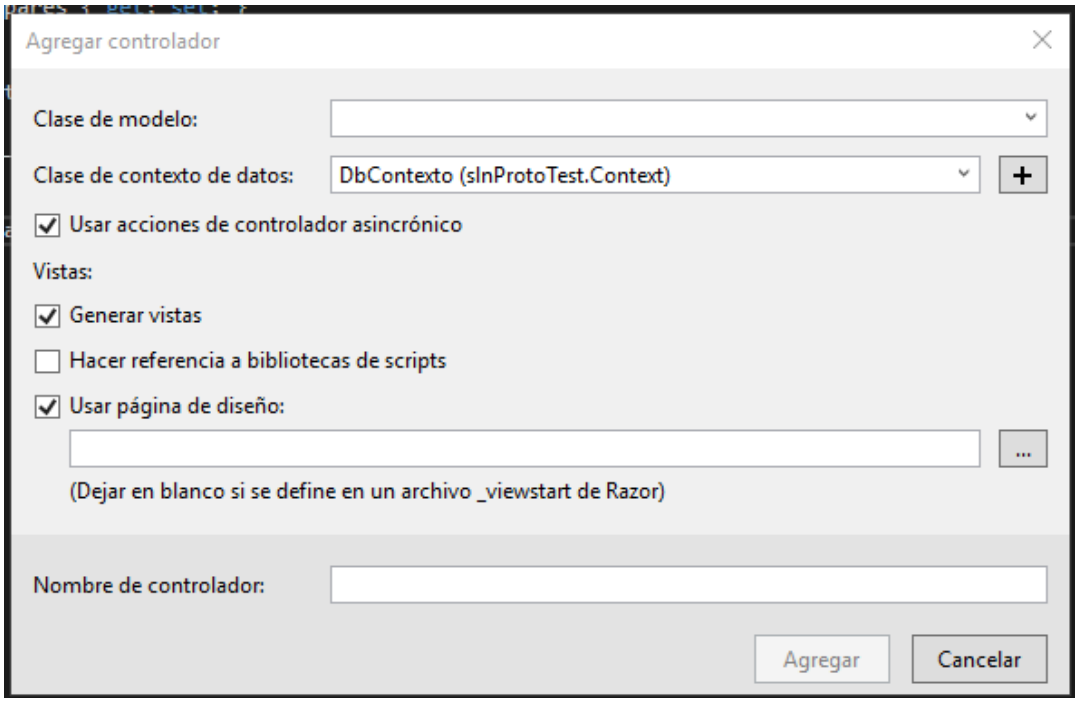


Figura 11. Definición de agregación de controlador MVC

Dentro de la presente opción nos permite seleccionar el modelo, en el cual se desarrollará el controlador y la vista, para lo cual seleccionamos el contexto de la base de datos creada, y configurada a desarrollar, se debe habilitar “acciones de modo asíncrono”, para optimización de transacciones, y en caso de tener una página de diseño, simplemente seleccionamos el modelo desde la raíz del proyecto; y para finalizar, agregamos el nombre con el cual se identificará al controlador y al conjunto de vistas generadas.

Vale recalcar, que por medio del “*template*” implementado para la solución, viene incorporado un diseño de vista a modo “*Frame*” o marco de opciones y cabecera, para el cual dentro de este podremos realizar las diversas configuraciones necesarias para centralizar la información y navegación de uso de la app.

La vista para el controlador Home y Error, los mismos que muestran la información necesaria en caso del usuario requiera.

4.1.7. Desarrollo de servicios

Para el desarrollo de los servicios WEB, alojados dentro de la app web, se implementó el tipo de servicio SOAP, para lo cual dentro de las diversas funcionalidades y herramientas presentes dentro del “*“Framework” “ASP.NET”*”, se crea un tipo de servicio web con extensión “*.asmx*”, característico de este ““Framework””, dentro del cual nos permite desarrollar los diversos servicios como funciones dentro de la ampliación, permitiendo así poder obtener información o a su vez guardar información dentro del servidor de base de datos, volviendo así seguro el manejo de información de las aplicaciones . Para lo cual se tiene desarrollado los presentes servicios:

- **Servicio “*getAgendamiento*”**

  El presente servicio despliega la información de tareas programadas en función al id de usuario insertado dentro del servicio, permitiendo devolver el resultado de tareas en formato “*JSON*”, para poder implementar dicha información dentro de las aplicaciones que lo requieran.
- **Servicio “*getlistadoTratantes*”**

EL presente servicio despliega la información del listado de usuarios de tipo tratantes en formato “*JSON*”, en función a la validación de “*login*” al momento de identificar al rol.

- **Servicio “*getPacientes*”**

  El presente servicio permite recuperar la información de pacientes que poseen tareas agendadas en formato “*JSON*”, para la respectiva administración.

- **Servicio “*getPacientesparaTratante*”**

  El presente servicio permite recuperar la información de pacientes que poseen tareas agendadas en formato “*JSON*”, para la respectiva administración en función del “*id*” del usuario tratante asignado al mismo

- **Servicio “*insertEstPares*”**

  El presente servicio permite ingresar los datos generados por la aplicación “*RukusGame*”, específicamente para el juego “*Doble o nada*” las mismas que permitirá almacenar los datos de cada sesión de juego (cada vez que despeje completo o incompleto el ejercicio) para almacenarlo dentro del servidor SQL Server.

- **Servicio “*InsertEstSimonDice*”**

  El presente servicio permite ingresar los datos generados por la aplicación “RukusGame”, específicamente para el juego “*Simón Dice*” las mismas que permitirá almacenar los datos de cada sesión de juego (cada vez que despeje completo o incompleto el ejercicio) para almacenarlo dentro del servidor SQL Server.

- **Inserttiempoentrepares**

  El presente servicio permite ingresar los datos generados por la aplicación “*RukusGame*”, específicamente las características de tiempo de reacción entre pares, y la imagen cargada en cada par. Las mismas que permitirá almacenar los datos de cada sesión de juego (cada vez que despeje completo o incompleto el ejercicio para almacenarlo dentro del servidor SQL Server.

- **Servicio “insertTiempoReaccion”**

El presente servicio permite ingresar los datos generados por la aplicación “RukusGame”, específicamente las características de tiempo de reacción entre pares, y la imagen cargada en cada par. Las mismas que permitirá almacenar los datos de cada sesión de juego (cada vez que despeje completo o incompleto el ejercicio para almacenarlo dentro del servidor SQL Server.

- **Servicio “updateAgendamiento”**

  El presente servicio permite actualizar los datos del agendamiento, específicamente las fechas de inicio de la app, y fecha fin de la app permitiendo registrar el inicio y fin de las partidas en función de las estadísticas reales, para luego proceder a las evaluaciones pertinentes.

- **Servicio “validateLogin”**

  El presente servicio permite validar el usuario y contraseña, para lo cual, si existe el usuario y contraseña asignado, devolverá Id del usuario para que permita proseguir dentro de las aplicaciones en cuestión.

4.2. Desarrollo de la app móvil

4.2.1. Librerías y consumo de servicios Web

Para el desarrollo de la aplicación móvil, fue necesario implementar la librería de denominad “*KSOAP*” la cual permite dentro de las aplicaciones de Android Studio, para lo cual esta librería aloja una serie de métodos necesarios para poder consumir servicios de tipo “*SOAP*”, que se encuentran alojadas en la nube, dentro de la app web

Su funcionalidad se base en configurar la dirección de URL donde se encuentra alojado los servicios, para luego administrar el nombre del servicio a consumir, y recupera la respuesta dentro de la app, para lo cual podemos hacer uso de la información proveniente en formato “*JSON*”, de tipo “*string*”, o a su vez una única respuesta de tipo “*string*”.

4.2.2. Diseño Visual

Dentro de la aplicación de tratamiento de usuarios, se tiene varios diseños de tipo "*.xml*", los mismos que nos permiten desarrollar la funcionalidad de control, para lo cual está establecido de la siguiente manera:

- **Log in**

  Al igual que la aplicación "*RukusGame*", la presente actividad componentes que nos permiten ingresar nuestro usuario y contraseña para poder validar por medio de los servicios web el usuario a autentificar, vale recalcar que la presente aplicación es solo funcional para los usuarios de tipo administrador, médico o tratante; en caso de un usuario de tipo Paciente desee ingresar a la aplicación, mostrará un mensaje de alerta indicando que solo usuarios, con los roles especificados con anterioridad pueden ingresar dentro de la aplicación.
- **Menús**

  El presente diseño está establecido para poder mostrar dos diferentes "*push up menú*", como se muestra en la Figura 12 y la Figura 13, los mismos que contienen integrado la funcionalidad necesaria para poder redireccionar a la actividad pertinente para control, en este caso se tiene dos tipos de menú, el que muestra las actividades del paciente en cuestión, y el que nos permite llegar a la actividad que nos permite registrar la fecha inicio, fecha fin de la tarea, observación y estado de esta.

Figura 12. Menú Ver Itinerario.

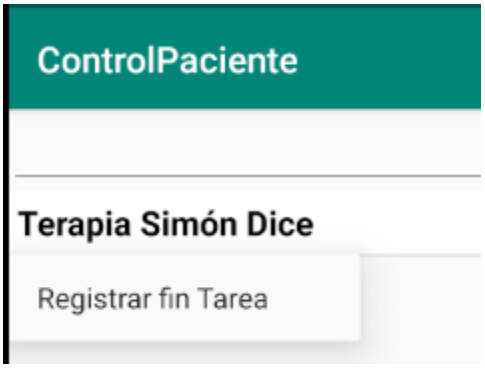


Figura 13. Menú Registrar Fin Tarea app "*ControlUsuario*"

- **Actividades alojadas con componente Listview**

  El presente diseño está establecido para poder visualizar la información recuperada por parte del servidor permitiendo así poder administrar los usuarios con agendamiento de tarea pertinente a tratar, en caso de ser usuario de tipo tratante simplemente se mostrará los usuarios asignados a su control para poder efectuar la actividad. Como se muestra en el Anexo 8, Figura 30.

  En el caso de seleccionar un usuario en específico, la segunda actividad mostrará la información referente a las actividades agendadas para dicho usuario, de tal manera que, que permite seleccionar la actividad por iniciar y registrar su debido fin, en función que le paciente cumpla o no pueda realizar la actividad. Como se puede apreciar en el Anexo 8, Figura 31.

- **Control de actividad**

  El presente diseño está establecido para poder realizar el control de inicio de tarea, control de fin de tarea, registrar el estado de la tarea, y por último algún tipo de observación dado el caso, para lo cual marcaría el fin del proceso de la actividad en cuestión en conjunto con la sesión de terapia establecida por el usuario de tipo médico. Como se muestra en la Figura 14.

Figura 14. Control Actividad app ControlPaciente

4.3. Adaptación aplicación "*RukusGame*"

Para la adaptación de la aplicación "*RukusGame*" para que funcione a la par con las aplicaciones web y móvil desarrolladas, y permita una integración a la plataforma de manera general, para lo cual, se implementó la librería de consumo de servicios "*KSoap*" la misma que permite consumir los servicios de la plataforma para poder almacenar la información a la base de datos, a la par de que cada vez que el usuario autenticado, o que este registrado dentro de la aplicación, entre a la apps e sincronice las tareas y muestre una notificación, de tipo "*push up*", como la que se muestra a continuación. Como se puede apreciar en el Anexo 8, la Figura 27.

4.4. Desarrollo de análisis de información

Para el análisis, y muestra final de la aplicación al usuario, se implementó la aplicación "*Power BI*", para lo cual fue estrictamente necesario desarrollar toda

la reportería en función de las estadísticas recopiladas en la base de datos, para lo cual se puede mostrar el progreso, en función del juego.

4.4.1. Desarrollo de informe de estadísticas de los juegos en general

En caso del juego como un reporte general con todos los usuarios, mostrando la información general del juego, en una gráfica de barras, con los datos generales del juego, en lado izquierdo, una gráfica de forma de dona, mastranto los colores, en función de aciertos, errores y no respondidos. En la esquina inferior izquierda, se agrega los filtros necesarios para poder obtener la información detallada en función del usuario. Para el caso peculiar de filtrado por agendamiento de tarea, se implementó una vista dentro de la base de datos, que permita integrar toda la información necesaria para poder adoptar los gráficos mencionados con anterioridad. Como se muestra en el Anexo 9, Figura 28

Para ambos juegos, poseen un informe estadístico en función a un alinea de tiempo, como parámetros establecidos se toman el rango de fechas de sesión por juego, a la par que puede evaluar el avance con una recta en función al tiempo, para lo cual se muestra los datos, como se muestra en el Anexo 9, Figura 29.

Para finalizar, se puede dentro de "*Power BI*" se desarrolla el modelo de vista de agendamiento de pacientes, en función al Calendario, de tal manera que facilita poder visualizar la disponibilidad del usuario paciente, médico o tratante, como se muestra en el Anexo 9, Figura 30.

Para lo cual cabe destacar que por medio de los controladores de filtro podemos seleccionar la información detallada, dependiendo del usuario, a la par que el reporte nos permite filtrar de mejor manera el calendario, en forma de Mes, Semana, Día o lista de tareas a realizar para una mejor funcionalidad el mismo.

4.5. Infraestructura de la aplicación Web y Base de datos en la nube

Para el presente proyecto, la infraestructura y recursos para funcionamiento del mismo fueron alojaos en Azure, por medio de la cuenta gratuita proporcionada por la universidad, permitiendo facilitar el uso de las aplicaciones, a la par de

poder centralizar el uso de la misma, de tal manera que, fue necesario crear un dentro de Azure el servicio "*Web App*", creando un grupo de recursos, el mismo que formará parte la aplicación, y la creación de un servidor de base de datos para alojar la información necesaria a impartir.

Para lo cual se llevó a cabo el siguiente proceso:

1. Dentro de la cuenta Azure, seleccionamos, dentro de general, seleccionamos la opción "*App Services*". Como se muestra en la Figura 15.

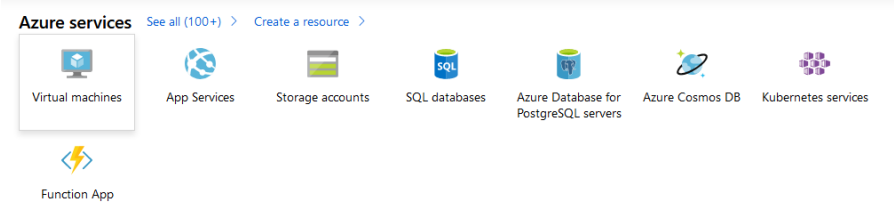


Figura 15. Servicios Disponibles Microsoft Azure

2. Una vez seleccionado, nos pedirá llenar los campos necesarios de la aplicación, para este caso es necesario seleccionar como Sistema operativo la opción *".NET Core 4.7*", y el plan que, en este caso, sería el plan "Free".
3. En el atributo de grupo de recursos, se debe seleccionar el grupo de recursos, para el óptimo desempeño de la aplicación, para lo cual, en el caso particular del presente proyecto se generó un nuevo grupo de recursos, denominado "*plataformamed*". Como se muestra en la Figura 16.

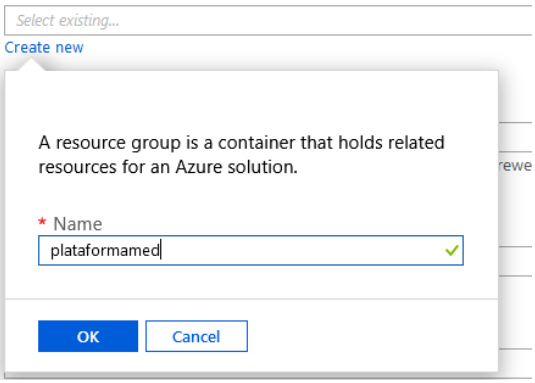


Figura 16. Creación de nuevo grupo de recursos

4. Una vez creado el grupo de recursos, se procede a seleccionar el plan a implementar dentro del mismo, para lo cual es estrictamente necesario seleccionar el tipo de plan F1, el mismo que es completamente gratuito. Como se muestra en la Figura 17.

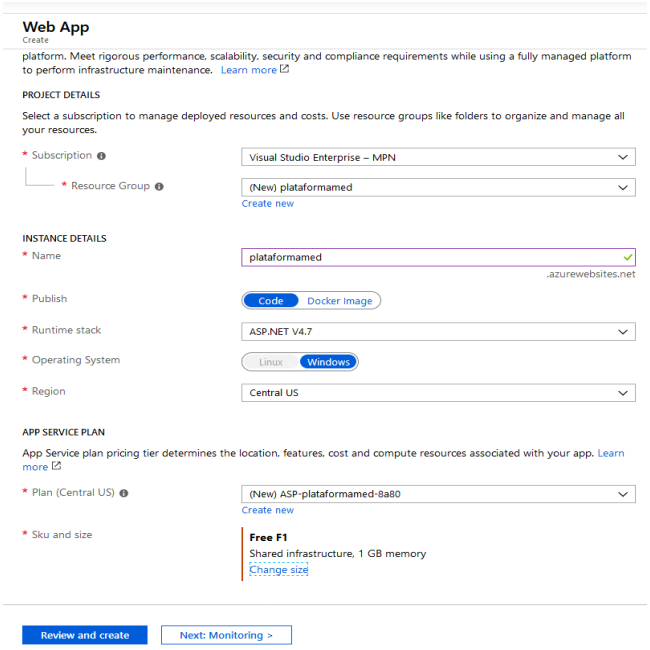


Figura 17. Selección de plan y finalización de configuraciones

5. Una vez finalizado la creación del "*App Service*" se procede a crear una base de datos de tipo SQL, para lo cual nos exigirá seleccionar el servidor donde queremos alojar la base de datos en cuestión o simplemente crear un nuevo servidor. Como se muestra en la Figura 18.

Figura 18. SQL Databases

Tomado de Microsoft Azure (2019)

6. En el caso del presente proyecto es necesario crear un nuevo servidor de base de datos, generando las configuraciones del caso tales como: asignación de la cuenta administrativa y su respectiva cuenta a la par del

nombre de la base de datos, y el plan de almacenamiento físico a almacenar, para este caso se seleccionó el plan básico de almacenamiento. Como se muestra en la Figura 19.

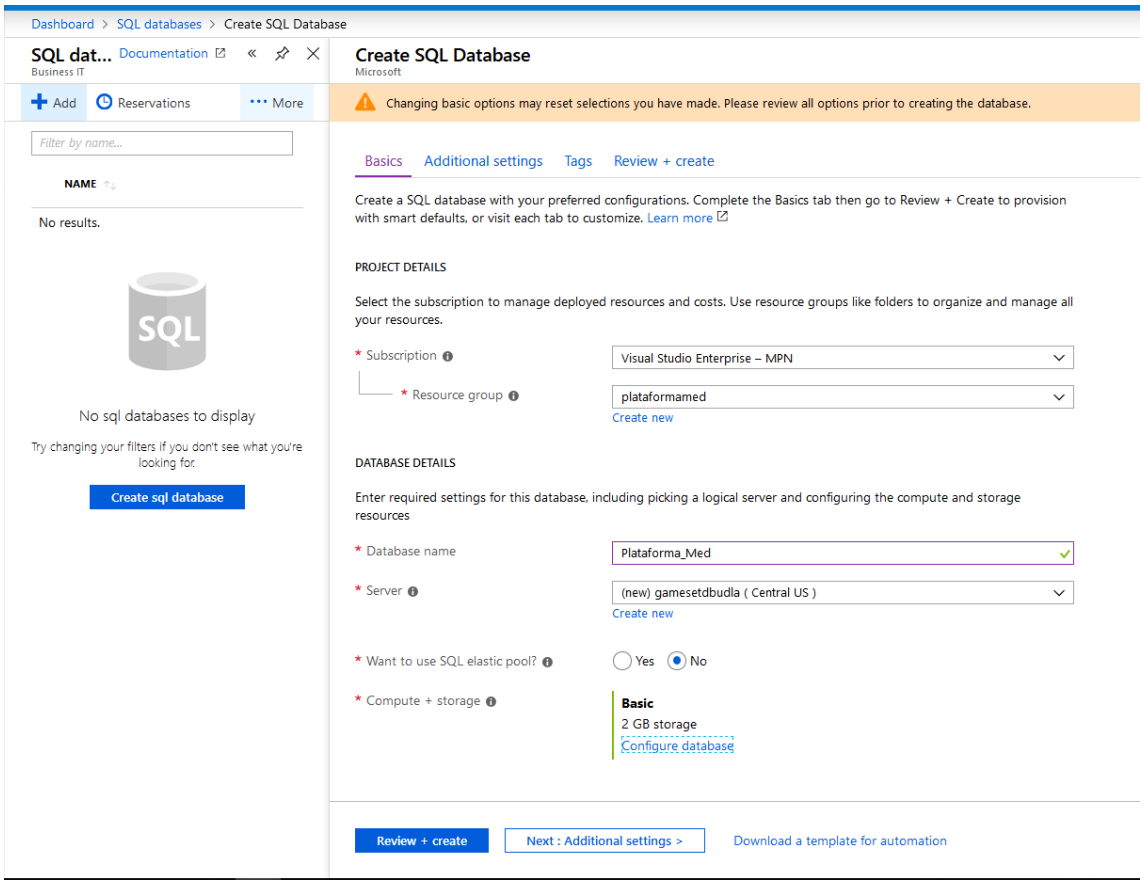


Figura 19. Creación de base de datos, con sus respectivos atributos.

5. CAPITULO V. Resultados.

5.1. Funcionamiento App Web "*PlataformaMED*"

Tras el diseño extenso de la aplicación Web "*PlataformaMed*", se puede detallar el funcionamiento básico de la aplicación, para lo cual está completamente definido en función al inicio de sesión de usuario a tratar, como se aprecia en la Figura 20, la misma delimitará toda función de uso administrativo para los usuarios que no posean el rol "*Administrador*", dentro de este podrá gestionar nuevos roles a crear, creación de usuarios de con el rol administrador, médico, paciente y tratante; a la par, de creación de nuevos roles y actividades.

Figura 20. Login app Web "*PlataformaMED*"

Una vez iniciada la sesión se puede crear un nuevo usuario por medio de los diversos CRUDS, seleccionando en la barra lateral izquierda la opción necesaria a tratar, en caso para los usuarios de tipo médico, solo podrán crear usuarios de tipo tratante y paciente. Como se aprecia en la Figura 21.

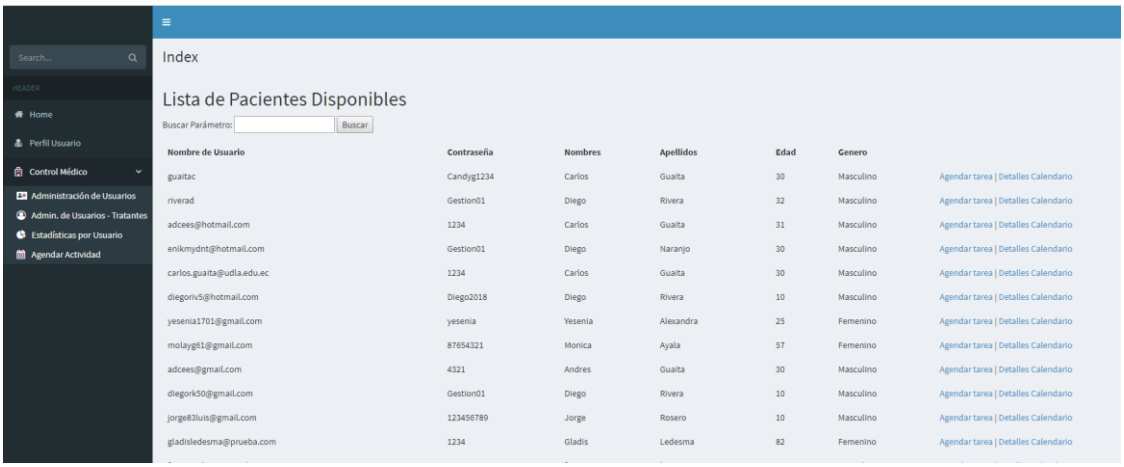


Figura 21. CRUD Usuarios.

Una vez obtenida la información del funcionamiento de la aplicación "*RukusGame*", posterior a la creación de la presente aplicación, a la par de generados roles y actividades, se puede hacer uso de la presente funcionalidad para poder agendar una tarea, como se parecía en la Figura 22, asignando una

fecha inicio, y fecha fin, que por medio de la aplicación “*ControlPaciente*”, se podrá recuperar y administrar desde el dispositivo móvil dicha tarea asignada al usuario de tipo paciente dentro de la aplicación “*RukusGame*”.

Figura 22. CRUD Listado tratantes.

Dentro de la funcionalidad del CRUD tratante, permite al médico asignar a los usuarios de tipo tratante a los diversos pacientes a disposición, para lo cual en primera instancia se mostrará la información del usuario a asignar al tratante, para una vez seleccionado el usuario, poder asociar a la persona que puede efectuar el tratamiento de la terapia.

Dentro del agendamiento de la tarea, asignar al usuario tratante para una tarea específica, en función de este registro previamente realizado.

La plataforma móvil presenta un módulo de reportes, el cual muestra la información estadística sobre las puntuaciones que realizo el paciente en cada una de las actividades efectuadas, en este caso, “*Simón Dice*” y “*Doble o Nada”,* tal como se puede apreciar en la Figura 24.

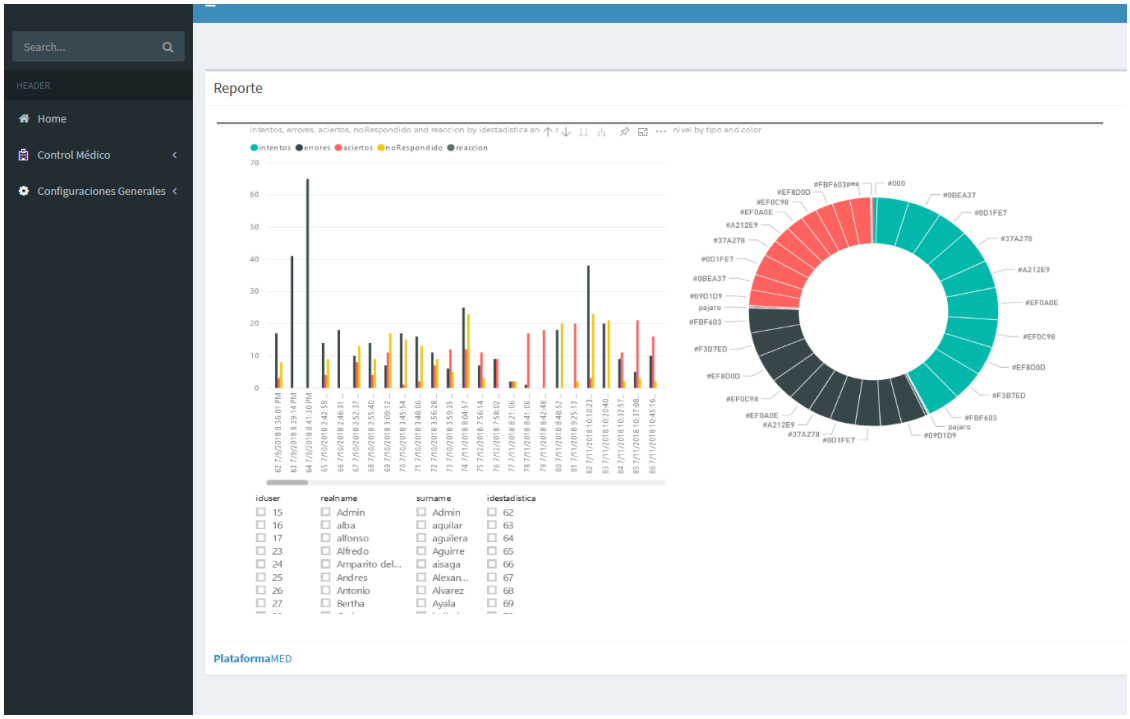


Figura 23. Presentación de Reportería

5.2. Funcionamiento App Móvil "*ControlPaciente*"

Tras el desarrollo de la aplicación "*ControlPaciente*", se logró obtener como resultado una aplicación móvil que, permite al usuario médico, o tratante, registrar el inicio y fin de una terapia asignada al usuario paciente, para lo cual permite actualizar toda la información de la base de datos referente a la tarea asignada, permitiendo obtener las métricas necesarias de evaluación, en función a la cantidad de sesiones generadas por la aplicación "*RukusGame*", para la respectiva evaluación objetiva y análisis de la información obtenida. Permitiendo lograr dicha funcionalidad por medio del uso de los diversos servicios web tipo SOAP, desarrollados e integrados dentro de la aplicación web "*PlataformaMed*", facilitando la centralización de la información.

Por medio de "*ListViews*" los mismos que albergan la información recibida por formato de tipo "*JSON*", permitieron administrar de mejor manera los pacientes y las tareas necesarias, a la par de una vista dedicada para el registro dentro de la base de datos del fin de la tarea, para poder visualizar toda la información dentro de los reportes de "*Power BI*". Como se puede apreciar en el Anexo 8, Figura 26 y Figura 27.

5.3. Funcionamiento App Móvil *“RukusGame”*

La aplicación generada con anterioridad como proyecto de la materia “*Aplicaciones móviles*” de la carrera de Electrónica y Redes, generar un prototipo de herramienta que permita mejorar el estilo de vida del adulto mayor, no solo generando terapias para la mejoría en salud, también permitió integrar a todo este grupo de personas que, por la edad, se desplazaban de la sociedad por no poder implementar o usar este tipo de dispositivos que, a la larga, facilitan la vida de todo individuo.

Para la integración y producto final de aplicación, se adaptó la misma para poder consumir los servicios Web de tipo “*SOAP*”, facilitando así la integración general con las otras aplicaciones desarrolladas para el presente proyecto, centralizando el uso de la información, a la par de permitiendo integrar una funcionalidad necesaria para poder alertar al usuario cuando debe iniciar la sesión por medio de una notificación “*push up*”. Como se muestra en la Anexo 8, Figura 28.

La estructura, funcionamiento y lógica del prototipo, está ambientado según el libro “*Estimulación cognitiva para adultos: guía básica*”, para lo cual todo funcionamiento de repetición, y ejercicios desarrollados según la lógica de programación están orientados en cumplir actividades mencionadas dentro del presente libro, para mejorar atributos tales como: “*motricidad”,” coordinación mano – ojo”, “concentración”, “memoria”.* Cumpliendo así el alcance y objetivo principal, para el desarrollo del presente prototipo. (Sardinero Peña 2010).

6. Conclusiones y Recomendaciones

6.1. Conclusiones

Por medio del presente proyecto se puede llegar a concluir que, por medio de las diversas herramientas tecnológicas disponibles en la actualidad, es factible el desarrollo integral y funcional de diversas aplicaciones que aportan de manera integra a la sociedad, a tal punto que, permitan esquematizarse de manera escalable, para el futuro desarrollo dentro de estas.

Tras el desarrollo del presente proyecto se puede llegar a concluir que, por medio de una integración sólida y efectiva, sin importar el lenguaje implementado para el desarrollo de estas, sea distinto uno de otro.

La implementación de la aplicación web, y móvil orientadas para el manejo de agendamiento de actividades y control de paciente, se concluye que las soluciones cumplen de manera básica y funcional las necesidades planteadas para el desarrollo de estas.

Se puede llegar a determinar que, gracias a diversos servicios alojados en la nube, se puede integrar diversos servicios web, para poder llevar a cabo una convergencia de datos, facilitando la intercomunicación entre herramientas, y centralización de información para uso interno de las mismas.

Por medio de la implementación de servicios web, se puede tener un nivel de seguridad de información, ya que, el resultado del uso de este lo ejecuta directamente el servidor, no la aplicación, para lo cual se presenta este uso de manera transparente para el usuario.

Por medio de diversas librerías de código existentes, de uso libre para el desarrollador, es factible facilitar y optimizar recursos al momento de desarrollo de cualquier aplicación, dado el caso como la plantilla *“AdminLTE*” para visual Studio, y librería “*KSoap*” para el consumo de servicios web dentro de las aplicaciones móviles.

Por medio de la reportería desarrollada en el presente proyecto, desarrollado dentro de la herramienta “*Power BI*”, se concluye que es factible mostrar de

manera ordenada y eficiente la información recopilada de diversas fuentes, permitiendo así, un fácil entendimiento para cualquier usuario final.

El desarrollo de la aplicación móvil "*Rukus Game*", logró el resultado requerido de terapia para mejoramiento de la matriz cognitiva del adulto mayor, gracias a las diversas tareas y actividades programadas entro de este, que cumple los estándares citados en el libro "Estimulación cognitiva para adultos: guía básica" referenciado en la presente investigación.

Finalmente, se puede definir que el presente proyecto puede ser considerado estable, permitiendo así a futuros estudiantes tomarlo como punto de partida para el desarrollo de un tema de titulación, debido a la gran complejidad y posibilidades de integración que permite el mismo.

6.2. Recomendaciones

Desarrollar futuras aplicaciones por medio de la herramienta Visual Studio, para lo cual facilitaría la integración de estas soluciones desarrolladas para cualquier entorno codificado con lenguaje "*C#*".

Tras el desarrollo paulatino dentro del estilo de programación "*Code First*" es necesario delimitar de manera óptima todo atributo y modelo necesario para poder proceder a la siguiente etapa de desarrollo, en el caso particular de aplicaciones "*"ASP.NET"*".

En caso de tomar el presente proyecto como punto de partida para el desarrollo o continuación de este, es aconsejable que el implementador mantenga la presente herramienta de desarrollo "Visual Studio" para poder integrar toda funcionalidad presentada dentro del mismo.

Es aconsejable que, para el caso de las aplicaciones móviles, tener en cuenta el tipo de documentación y funcionamiento que lleve a cabo la última versión de Android presente el mercado.

Es aconsejable implementar herramientas de gestión de base de datos, para facilitar el manejo, uso y administración de estas, para lo cual se puede llevar acabo por medio de la herramienta "*Navicat*".

Dentro del desarrollo de aplicaciones móviles dentro del "*"Framework"*" de "*Android Studio*", es recomendable validar si el error se presenta dentro del "*"Framework"*", y no dentro del código, para poder tomar cualquier medida correctiva dentro de este.

REFERENCIAS

Alonso, Gustavo, et al. "*Web services*." Web Services. Springer, Berlin, Heidelberg, 2004. 123-149.

Android Developers (2019). *Conoce Android Studio*. Recuperado el 2 de junio del 2019 de: https://developer.android.com/studio/intro?hl=es-419

Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual.

IBM (2018). *What is a web service?.* Recuperado el 2 de junio del 2019 de: https://www.ibm.com/support/knowledgecenter/SSGMCP_4.2.0/com.ibm.cics.ts.webservices.doc/concepts/dfhws_definition.html

JSON (2019). *Introducing JSON*. Recuperado el 2 de junio del 2019 de: https://www.json.org/

Oracle (2019). *What Are RESTful Web Services?.* Recuperado el 2 de junio del 2019 de: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

Microsoft Documents (2019). *Introduction to the C# Language and the .NET "Framework".* Recuperado el 2 de junio del 2019 de: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-”Framework”

Microsoft Documents (2019). *Language Integrated Query (LINQ).* Recuperado el 2 de junio del 2019 de: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-”Framework”

Microsoft Documents (2019). *Get started with the .NET "Framework"*. Recuperado el 2 de junio del 2019 de: https://docs.microsoft.com/en-us/dotnet/”Framework”/get-started/index

Microsoft Documents(2019). Introduction to “ASP.NET” Core?. Recuperado el 2 de junio del 2019 de:https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.2

Microsoft Documents (2019). *Get Started with Entity “Framework” 6 Code First using MVC 5.* Recuperado el 2 de junio del 2019 de: https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-”Framework”-data-model-for-an-asp-net-mvc-application

Microsoft Documents (2019). Get Started with Entity “Framework” 6 Code First using MVC 5. Recuperado el 2 de junio del 2019 de: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.2

MySQL Developers (2019). *MySQL 8.0 Reference Manual.* Recuperado el 2 de junio del 2019 de: https://dev.mysql.com/doc/refman/8.0/en/introduction.html

Navicat Premium (2019). *Navicat Premium.* Recuperado el 2 de junio del 2019 de: https://www.navicat.com/es/products/navicat-premium

Power BI (2019). *What is Power BI?.* Recuperado el 2 de junio del 2019 de: https://powerbi.microsoft.com/en-us/what-is-power-bi/#

Rees, Dayle, and Antonio Laguna. "*Laravel: Code Happy (ES)*." (2012).

Sardinero Peña, A. (2010). Estimulación cognitiva para adultos: guía básica. Gesfomedia. Madrid.

ANEXOS

Anexo 1: Código ejemplo para consumo de servicios web tipo SOAP dentro de Android Studio.

```
private class ParseTask extends AsyncTask<Void, Void, String> {

        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        String resultJson = "";

        @Override
        protected String doInBackground(Void... params) {
            try {
                request = new SoapObject(NAMESPACE, METHOD_NAME1);
                String idtarea = getIntent().getExtras().getString("idtarea");
                request.addProperty("id", idtarea);
                request.addProperty("estadotarea",est);
                request.addProperty("observaciones",observaciones);
                request.addProperty("fechafin",inputFormat.format(now));
                SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
                envelope.dotNet = true;
                envelope.setOutputSoapObject(request);
                HttpTransportSE httpTransport = new HttpTransportSE(SOAP_URL);
                try {
                    httpTransport.call(SOAP_ACTION1, envelope);
                    SoapPrimitive fahtocel = (SoapPrimitive) envelope.getResponse();
                    rslt=fahtocel.toString();
                    Log.wtf("Actualización de datos", fahtocel.toString());
                } catch (Exception e) {
                    e.getMessage();
                }

                //resultJson = rslt.toString();
                Log.d("FOR_LOG", rslt);

            } catch (Exception e) {
                e.printStackTrace();
            }
            return rslt;
        }


        protected void onPostExecute(String strJson) {
            super.onPostExecute(strJson);


            Toast.makeText(getApplicationContext(),  "Carga Completada", Toast.LENGTH_SHORT).show();
            try {

                Log.d("My App", rslt);

            } catch (Throwable t) {
                Log.e("My App", "Error al cargar los datos: \"" + rslt + "\"");
            }

        }
    }
```

Anexo 2: Código ejemplo para mostrar notificaciones push up dentro de las aplicaciones de AndroidStudio

```
protected void displayNotification(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        // Create the NotificationChannel
        NotificationChannel channel1 = new NotificationChannel(
                CHANNEL_1_ID,
                "Channel 1",
                NotificationManager.IMPORTANCE_HIGH
        );
        channel1.setDescription("This is Channel 1");
        NotificationManager manager =
getSystemService(NotificationManager.class);
        manager.createNotificationChannel(channel1);
    }

    Intent i = new Intent(this, recordatorio.class);
    i.putExtra("notificationID", notificationID);

    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
i, 0);
    NotificationManager nm =
(NotificationManager)getSystemService(NOTIFICATION_SERVICE);
    CharSequence ticker ="RukusGame";
    CharSequence contentTitle = "Tarea Pendiente";
    CharSequence contentText = "Ud tiene una terapia de tipo: simon
dice";

    Notification noti = new NotificationCompat.Builder(this,
CHANNEL_1_ID)
            .setContentIntent(pendingIntent)
            .setTicker(ticker)
            .setContentTitle(contentTitle)
            .setContentText(contentText)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .addAction(R.drawable.ic_launcher_background, ticker,
pendingIntent)
            .setVibrate(new long[] {100, 250, 100, 500})
            .build();
    nm.notify(notificationID, noti);
}
```

Anexo 3: Inserción de reportería dentro de la plataforma “ASP.NET”

```
<div class="box">
    <div class="box-header with-border">
        <h3 class="box-title">Reporte</h3>
    </div>
    <div class="box-body">
        <iframe width="1140" height="541.25"
src="https://app.powerbi.com/reportEmbed?reportId=1adc8066-3d1d-4e47-be9d-
9e50dae625cc&autoAuth=true&ctid=24884996-6863-4925-a1ba-f1a160b581e2"
frameborder="0" allowFullScreen="true"></iframe>
    </div>
    <!-- /.box-body -->
    <div class="box-footer">
        <a><b>Plataforma</b>MED</a>
    </div>
    <!-- /.box-footer-->
</div>
```

Anexo 4: Función Create (OnPost) personalziada para el agendamiento de tarea.

```
public async Task<ActionResult> Create([Bind(Include =
"idtarea,idactividad,descripcion,fechainiciotarea,fechafintarea")] agendamientotarea agendamientotarea)
        {
           ModelState state;

}
           if (ModelState.IsValid)
           {

                agendamientotarea.estadotarea = "Pendiente";
                agendamientotarea.observaciones = "";
                agendamientotarea.fechainicioapp = agendamientotarea.fechainiciotarea;
                agendamientotarea.fechafinapp = agendamientotarea.fechafintarea;
                db.agendamientotarea.Add(agendamientotarea);
                await db.SaveChangesAsync();
                agendamientotarea lastid = db.agendamientotarea.OrderByDescending(u =>
u.idtarea).FirstOrDefault();
                int idtarealast = lastid.idtarea;
                // almacenar realicon paciente - ficha medica
                agendamientotarea_user agendamiento_user = new agendamientotarea_user();
                agendamiento_user.iduser = int.Parse(Session["idUsuario"].ToString());
                agendamiento_user.idtarea = idtarealast;
                db.agendamientotarea_uer.Add(agendamiento_user);
                await db.SaveChangesAsync();
                // almacenar realacion medico - ficha medica
                agendamientotarea_user agendamiento_medic = new agendamientotarea_user();
                agendamiento_medic.iduser = int.Parse(Session["idMedico"].ToString());
                agendamiento_medic.idtarea = idtarealast;
                db.agendamientotarea_uer.Add(agendamiento_medic);
                await db.SaveChangesAsync();
                return RedirectToAction("Index",new { id=Session["IdUsuario"]});

            }

            ViewBag.idactividad = new SelectList(db.actividad, "idactividad", "nombreactividad",
agendamientotarea.idactividad);
            return View(agendamientotarea);
        }
```

Anexo 5: Eliminación de pluralización al momento de generar la base de datos por medio del estilo de programación “Code First”

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {

            //no exista pluralización

            modelBuilder.Entity<user>().ToTable("user");

            modelBuilder.Entity<FichaMedica>().ToTable("fichamedica");

            modelBuilder.Entity<estadisticapares>().ToTable("estadisticapares");

            modelBuilder.Entity<estadisticasimondice>().ToTable("estadisticasimon
            dice");

            modelBuilder.Entity<tiempoentrepares>().ToTable("tiempoentrepares");

            modelBuilder.Entity<tiemporeaccion>().ToTable("tiemporeaccion");
                        modelBuilder.Entity<actividad>().ToTable("actividad");

            modelBuilder.Entity<agendamientotarea>().ToTable("agendamientotarea")
            ;

            modelBuilder.Entity<agendamientotarea_user>().ToTable("agendamientota
            rea_user");

            modelBuilder.Entity<fichamedica_user>().ToTable("fichamedica_user");

            modelBuilder.Entity<listatratantes>().ToTable("listatratantes");
            modelBuilder.Entity<Roles>().ToTable("roles");


                  }
```

Anexo 6: Entidades aplicación web

Entidad Actividad

Tabla 1. Modelo tabla Actividad

Atributo	**Tipo Variable**	**DataAnotations**
idactividad	int	Key
nombreactividad	string	Display(Name = "Nombre de la Actividad")
tipo	string	Display(Name = "Tipo de la Actividad")
agendamientotarea	ICollection<agendamientotarea>	

Entidad AgendamientoTarea

Tabla 2. Modelo de la tabla AgendamientoTarea

Atributo	**Tipo Variable**	**DataAnotations**
idatarea	int	Key
idactividad	int	
actividad	actividad	
descripcion	String	[Display(Name = "Descripcción")]
fechainiciotarea	DateTime	[Display(Name = "Fecha Inicio")]

fechafintarea	DateTime	[Display(Name ="Fecha Fin")]
fechainicioapp	DateTime	[Display(Name = "Fecha Inicio App")]
fechafinapp	DateTime	[Display(Name = "Fecha Fin App")]
estadotarea	String	[Display(Name ="Estado de la tarea")]
observaciones	String	[Display(Name ="Observación ")]
Agendamientotarea_User	ICollection<agendamientotarea_user>	

Entidad Agendamientotarea_user

Tabla 3.Modelo de la tabla Agendamientotarea_user

Atributo	**Tipo Variable**	**DataAnotations**
idagendamiento_tarea	int	Key
idtarea	string	
Agendamientotarea	agendamientotarea	
iduser	int	
User	user	

Entidad Estadisticapares

Tabla 4.Modelo de la tabla Estadisticapares

Atributo	**Tipo Variable**	**DataAnotations**
idestadistica	int	Key
totalTimeMemoryGame	int	[Display(Name ="tiempto total de juego")]
numberMistakes	int	[Display(Name ="Número de pares")]
iduser	int	[Display(Name ="Usuario")]
User	user	
sesion	DateTime	[Display(Name ="Fecha de sesión")]
nivel	Int	[Display(Name ="Dificultad")]

Estadisticasimondice

Tabla 5. Modelo de la tabla Estadisticasimondice

Atributo	**Tipo Variable**	**DataAnotations**
idestadistica	int	Key
iduser	int	

User	user	
iduser	int	
User	user	
aciertos	int	[Display(Name ="Aciertos Realizados")]
errores	Int	[Display(Name ="Errores Realizados")]
noRespondido	int	[Display(Name ="Acciones no respondidas")]
reaccion	Int	[Display(Name ="Tiempo de reacción entre colores")]
intentos	int	[Display(Name ="Número de intentos")]
session	DateTime	[Display (Name ="Fecha de sesión")]
Tiemporeaccion	ICollection<tiemporeaccion>	

Entidad FichaMedica

Tabla 6. Modelo de la tabla FichaMedica

Atributo	Tipo Variable	DataAnotations
idFichaMedica	int	Key
PadecimientosMedicos	string	[Display(Name = "Padecimientos Médicos")]
NotasMédicas	string	[[Display(Name = "Notas del Médico")]
AlergiasReacciones	int	Display(Name = "Alérgias o Reacciones")]
Medicamentos	string	[Display(Name = "Medicamentos")]
tipoSangre	string	[Display(Name ="Tipo de Sangre")]
Donante	bool	[Display(Name ="Es donante")]
Peso	string	
Estatura	string	
fichamedica_User	ICollection<fichamedica_user>	

FichaMedica_user

Tabla 7. Modelo de la tabla FichaMedica_user

Atributo	Tipo Variable	DataAnotations
idfichamedica_user	int	Key

Iduser	Int	
User	user	
idFichamedica	idFichamedica	
FichaMedica	FichaMedica	

Listatratantes

Tabla 8. Modelo de la tabla Listatratantes

Atributo	**Tipo Variable**	**DataAnotations**
idusuariotratante	int	Key
iduser	Int	[Display(Name ="Usuario encargado.")]
User	user	
Idmedic	Idmedic	
Idpaciente	idpaciente	

Roles

Tabla 9. Modelo de la tabla Roles

Atributo	**Tipo Variable**	**DataAnotations**
idrol	int	Key
Nombre	string	[Display(Name ="Usuario encargado.")]
User	ICollection<user>	

Tiempoenterpares

*T*abla 10.

Modelo de la tabla Tiempoentrepares

Atributo	**Tipo Variable**	**DataAnotations**
idtiempopares	Int	Key
idestadistica	Int	
estadisticapares	estadisticapares	
timeBetweenPairs	int	[Display(Name ="Tiempo entre Pares")]
Imagen	string	

Tiemporeaccion

Tabla 11.

Modelo de la tabla Tiemporeaccion

Atributo	**Tipo Variable**	**DataAnotations**
Idreaccion	int	Key
idestadistica	int	
estadisticasimondice	estadisticasimondice	
tiempo	int	[Display(Name = "Tiempo total de juego")]
Nivel	string	[Display(Name = "Dificultad")]
color	string	[Display(Name = "Color")]
Tipo	string	[Display(Name = "Correcto - Incorrecto")]

User

Tabla 12. Modelo de la Tabla User.

Atributo	**Tipo Variable**	**DataAnotations**
iduser	int	Key
username	string	[Display(Name = "Nombre de Usuario")]
password	string	[Display(Name = "Contraseña")]
realname	string	[Display(Name = "Nombres")]
surname	string	[Display(Name = "Apellidos")]
rol	int	[ForeignKey("rol")]
Roles	roles	
edad	int	[Display(Name = "Edad")]
genero	int	[Display(Name = "Genero")]
estadsiticapares	ICollection<estadisticapares>	
estadisticasimondice	ICollection<estadisticasimondice>	
fichamedica_user	ICollection<fichamedica_user>	
agendamientotarea_user	ICollection<agendamientotarea_user>	

Anexo 7: Diagrama Entidad relación de base de datos “PlataformaMed_db”

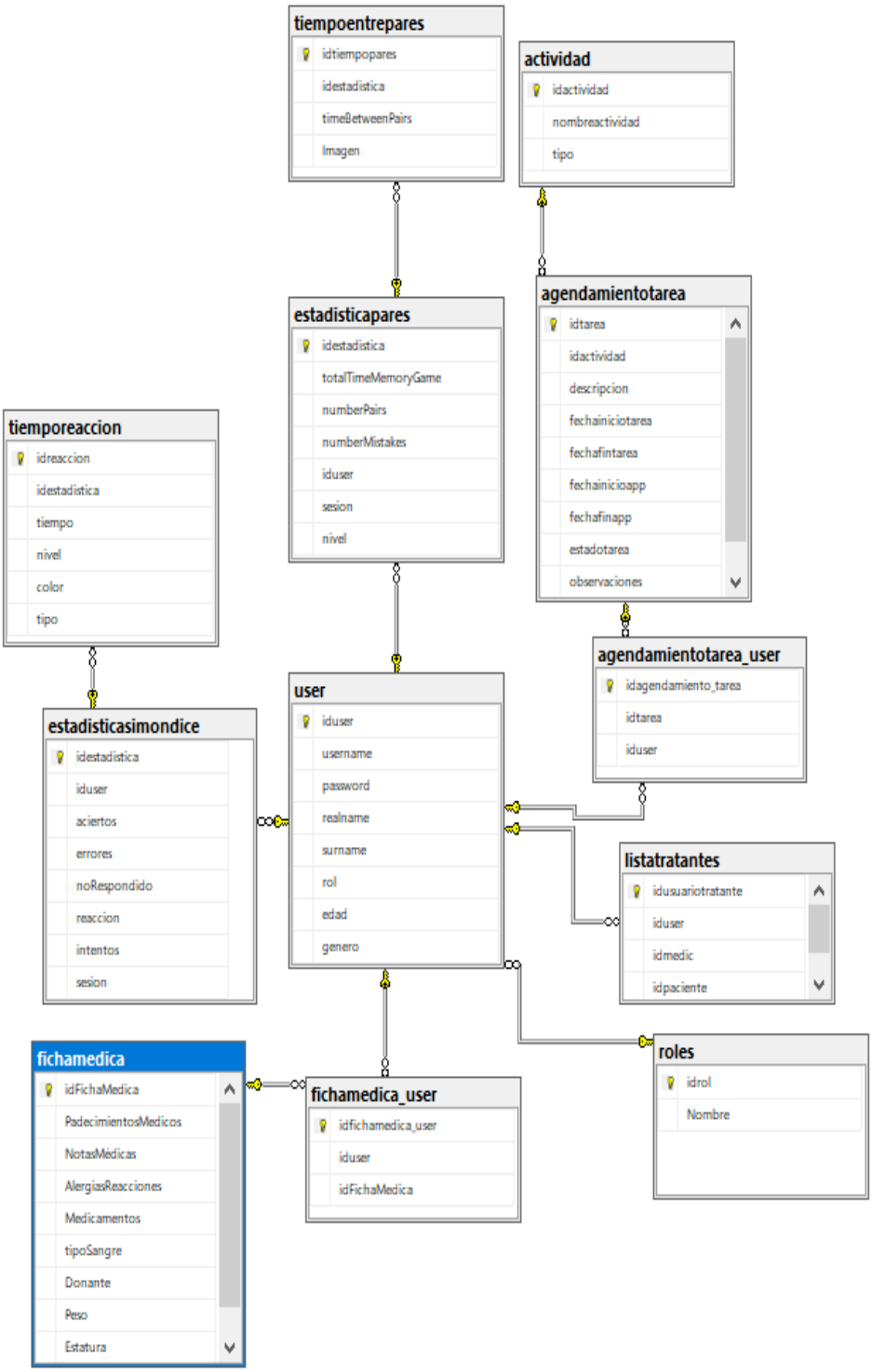


Figura 24. Diagrama Entidad relación de base de datos “*PlataformaMed_db*” (en base al modelo anterior).

Anexo 8: Capturas de pantalla Aplicación "*ControlPaciente"*

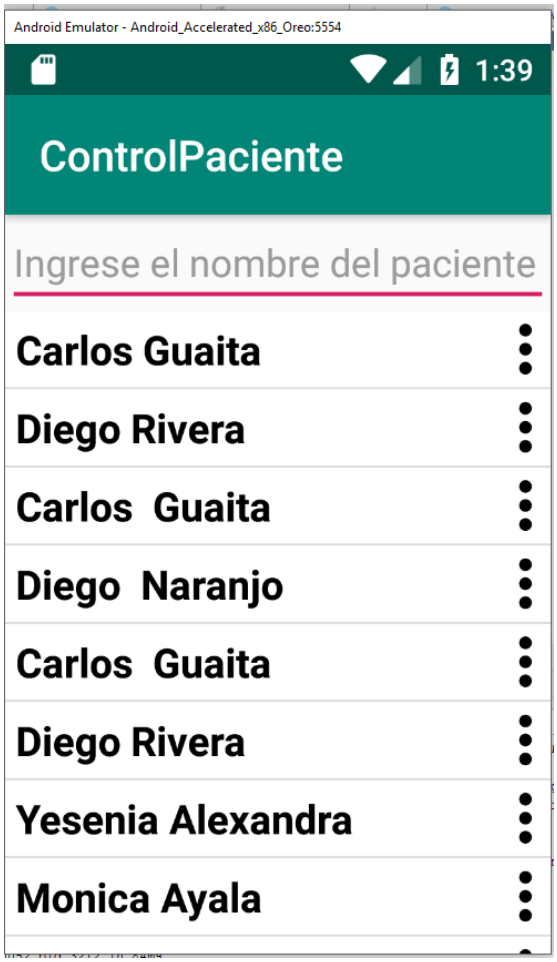


Figura 25. Lista Usuarios app ControlPaciente.

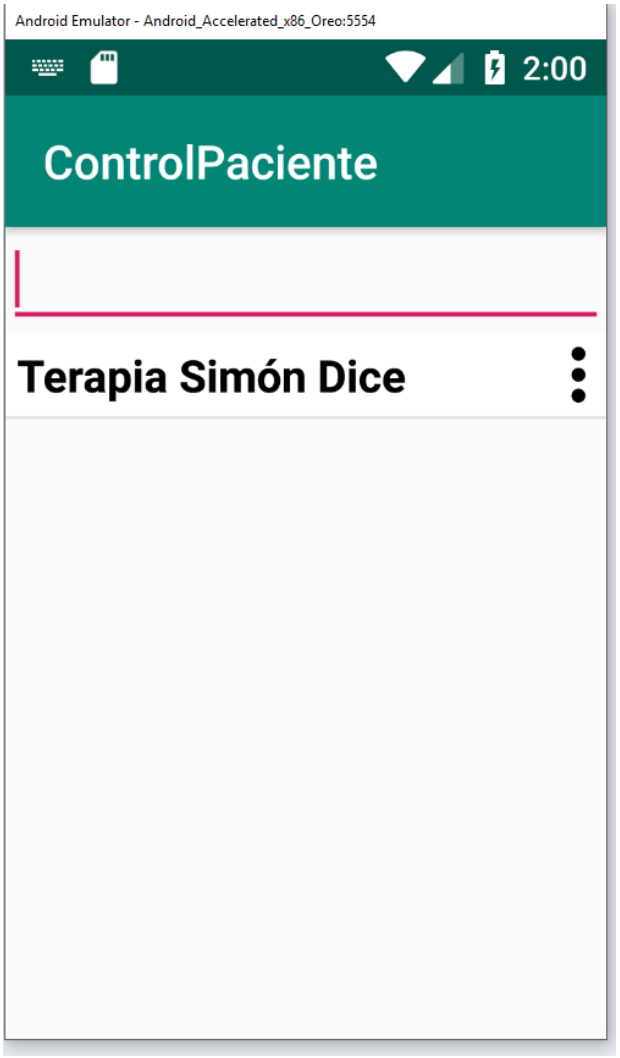


Figura 26. Lista de tareas app "ControlPaciente"

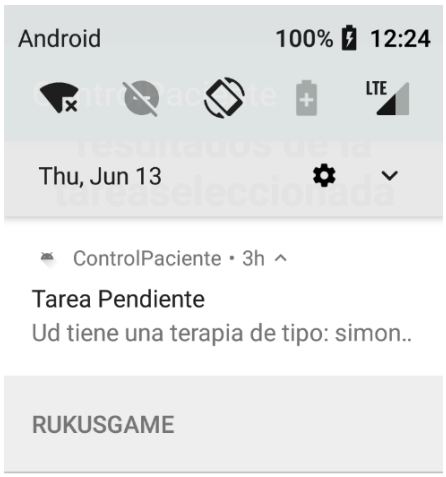


Figura 27. Notificación de Juego.

Anexo 9: Reportes Estadísticos "*Power BI*"

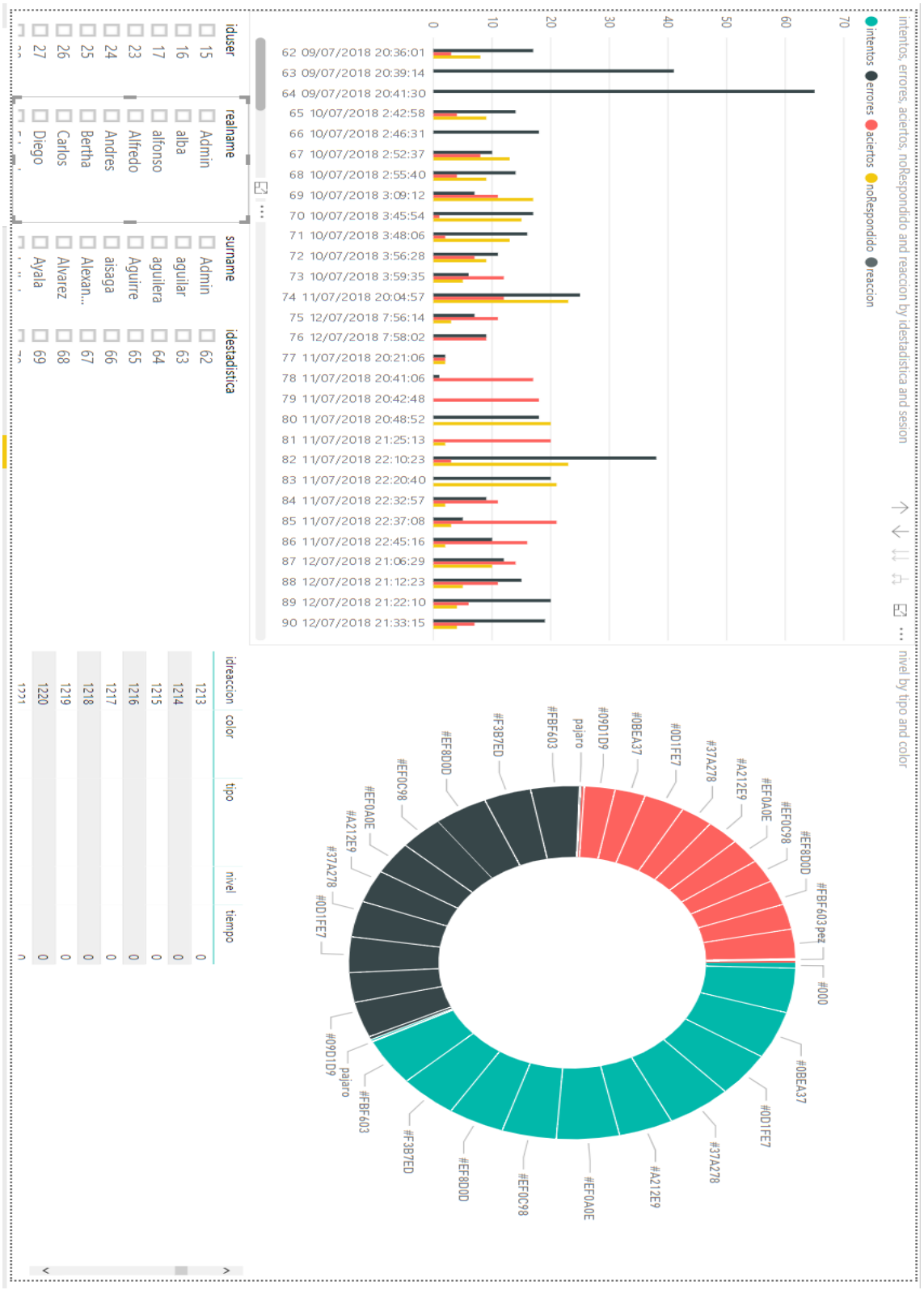


Figura 28. Reporte de estadísticas Simón dice, tanto por filtrado de agendamiento como vista general.

Figura 29. Recta de sesiones en función al tiempo.

Figura 30. Calendario de disponibilidad pacientes.