Che mozo

Diego Sebastián Ginko

***Resumen*—Che mozo es un ecosistema de comunicación entre el comensal y el camarero, integrado por un dispositivo IOT ubicado en la mesa para uso del comensal, una aplicación móvil para el camarero y una aplicación a modo de servidor para oficiar de nexo entre ambos.**

***Palabras clave*—** Gastronomía, Internet of Things, MQTT, Operating systems, ESP8266, Protocols

I. Introducción

Che mozo inicia como un impulso de modernización en el sector gastronómico, haciendo foco en encontrar un método para agilizar la comunicación entre el comensal y el camarero. Valiéndonos de dispositivos que ya están en el mercado como ser los smartphones y dispositivos IOT [1] generamos un ecosistema expandible para establecer un nexo directo entre las partes, intentando que el costo económico de entrada sea el menor posible. Logrando una comunicación directa se espera modernizar la relación establecida por años entre el comensal y el camarero haciendo más fluido el contacto con las nuevas generaciones más avanzadas a la tecnología [2]. El objetivo principal es proveer de un conjunto de herramientas y soluciones accesibles que agilicen el desarrollo de la actividad gastronómica y la experiencia del cliente en el proceso. Las metas a cumplir son:

- Aprender y experimentar con dispositivos IOT.
- Mejorar el entendimiento en comunicaciones entre dispositivos.
- Incentivar a la mejora y modernización del rubro gastronómico a través de la tecnología.

II. Caso de estudio

II-A. Situación

Vamos a situarnos en un restaurante, un local con sus plazas completamente ocupadas, en este contexto se propone como caso de estudio la interacción entre el comensal y un camarero. Ya sea para llamar su atención o para pedir la cuenta se utilizan una serie de gestos los cuales pueden variar dependiendo de la cultura local, además, los mismos sólo son válidos cuando el camarero está dirigiendo su atención al comensal y por la distancia entre los mismos utilizar el diálogo verbal no sería una opción. También se debe tener en cuenta la tendencia de las nuevas generaciones a limitar las interacciones sociales [2], ¿De qué manera podríamos valernos de la tecnología para facilitar esta comunicación?.

Una opción podría ser un aplicativo para celular, aprovechando la adaptación masiva de los mismos [3], esto se ve descartado ya que en principio, el comensal tendría que estar dispuesto a descargarse la aplicación, disponer de buena conexión para hacerlo y tener espacio disponible en su dispositivo.

Viendo las problemáticas de la primera opción se puede optar por algo que no requiere descarga, como una aplicación web. Las razones por las que se descarta esta posibilidad son las siguientes:

- Queda limitado a la conexión a internet del dispositivo para el envío y del local para la recepción.
- Las funciones del dispositivo a utilizar quedan limitadas a las accesibles por el navegador.
- Es necesario compatibilizar y probar con todos los navegadores disponibles.

Llegado a este punto la utilización del teléfono móvil del comensal ya no es viable, por lo que se opta por la posibilidad de utilizar un dispositivo propio, el cual ya se encuentre en el local. Este debe cumplir con lo siguiente:

- Ser simple de utilizar.
- Tener un tamaño adecuado para ser ubicado en una mesa y no ocupar mucho espacio (de esta forma no entorpece la experiencia del comensal).
- Tener un costo reducido.
- Disponer de conexión inalámbrica.

Para cumplir con la simpleza, se decidió utilizar dos botones físicos, uno para llamar al camarero y el otro para pedir la cuenta. En cuanto al tamaño y el costo, existen placas de desarrollo como arduino las cuales vienen en tamaños reducidos y son económicas, ya que la mayoría no cuentan con conexión inalámbrica se opta por las placas basadas en esp8266, las cuales tienen una programación similar, cuentan con más prestaciones y no elevan considerablemente el costo [4].

En el caso de los camareros debemos tener en cuenta que se encuentran en constante movimiento, por lo que un dispositivo fijo no tendría sentido alguno. En un principio, se establecen las necesidades para el uso del mismo:

- Disponer de conexión inalámbrica.
- Que sea portátil.
- Bajo consumo energético.
- Posibilidad de notificar de forma no sonora.
- Bajo costo.

Ante estos requerimientos es posible utilizar un dispositivo móvil con el que ya cuenten, lo que no incurrirá en un costo adicional. A través de una app pueden llegar las notificaciones de las activaciones de los distintos dispositivos de las mesas y notificar al camarero para que responda de forma acorde.

II-B. Comunicación entre dispositivos

Ya definidos los dispositivos queda establecer la forma de comunicación. Se opta por plantear un escenario en donde se encuentre un servidor que sea quien recibe las notificaciones de los dispositivos pertenecientes a los comensales y luego se las comunique a los dispositivos pertenecientes a los camareros, de esta forma se podrá realizar una configuración por zonas del salón y evitar que todas las notificaciones lleguen a todos los camareros.

II-C. Protocolo

Para tener una comunicación debemos tener un protocolo, es decir un conjunto de reglas que tanto el servidor como los dispositivos conozcan para poder establecer un diálogo. Para este caso vamos a utilizar MQTT debido a que está probado que funciona eficazmente en ámbitos con dispositivos IOT de baja potencia [5]. MQTT utiliza un modelo de Broker-Client donde los clientes pueden suscribirse a temas generados por otros clientes de esta forma se pueden interconectar nuestros dispositivos de una forma simple.

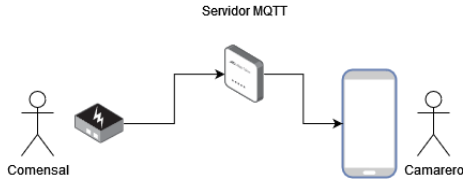
En la siguiente figura podemos ver un diagrama de como quedaría el ecosistema.

III. Marco teórico

III-A. Internet de las cosas

La tecnología IOT [1] (Internet of things o internet de las cosas) cuyo comienzo data del año 1980 [6] está instaurada en el mercado desde su aceptación masiva por parte de la industria entre los años 2008 y 2009 [7]. Conocemos a internet de las cosas como un sistema de dispositivos computarizados conectados entre sí, tanto máquinas

Figura 1. Esquema del ecosistema de comunicación

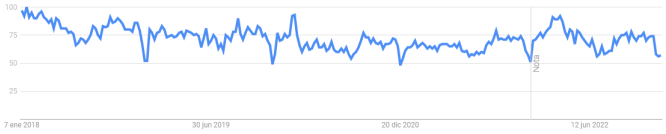


analógicas como digitales, los cuales disponen de un identificador único (UID) para ser reconocidos y con la capacidad de transmitir datos a través de una red sin intervención humana.

El término internet of things tiene su origen gracias a Kevin Ashton [8] aunque su intención era que fuese llamado internet para las cosas, él lo define como “Simplemente el punto en el tiempo donde más cosas u objetos estén conectadas a internet que personas”.

En la siguiente figura extraída de google trends, cuya escala es 100 siendo muy popular y 0 siendo poco popular podemos denotar el interés por el término en el periodo entre el 01/01/2018 y el 01/01/2023 [9]

Figura 2. Búsqueda del termino internet of things



III-B. *Placas de desarrollo*

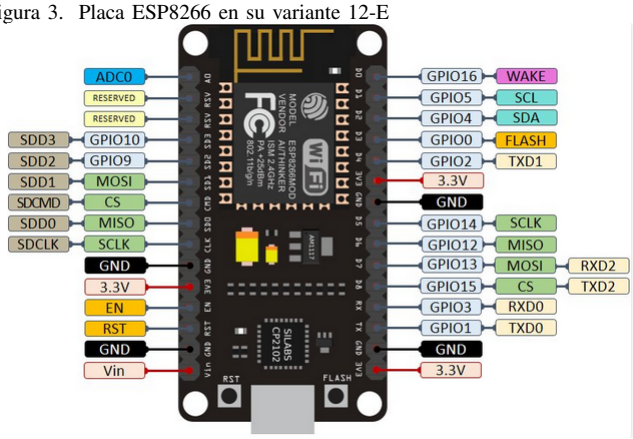
III-B1. Arduino: Arduino [10] es una plataforma de creación de electrónica de código abierto, está basada en hardware y software libre, flexible y fácil de usar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa otorgándoles diferentes tipos de uso mediante la programación con el entorno Arduino IDE. Arduino nace en 2003 como una alternativa más económica a las placas BASIC Stamp, y promueve la filosofía ’learning by doing’, utiliza microcontroladores ATMEL y permite conectar placas de expansión (shields) a través de sus puertos de entrada y salida, para agregar funcionalidades como sensores y módulos de comunicación de forma sencilla.

III-B2. Esp8266: Es un chip de bajo costo Wi-Fi con un stack TCP/IP completo y un microcontrolador, fabricado por Espressif [11], una empresa proveniente de Shanghái, China. El chip se popularizó en 2014 con el módulo ESP-01, dado que permitía a otros microcontroladores conectarse a una red inalámbrica Wi-Fi y realizar conexiones simples con TCP/IP. Integra un procesador de 32 bits Tensilica L106 de arquitectura RISC que funciona a una velocidad de 80 MHz, con una velocidad máxima de 160 MHz, también tiene integrados componentes como TR switch, balun, LNA, amplificador de potencia de RF y una red de adaptación de impedancias. Soporta autenticación WEP y WPA/WPA2 y tiene 16 pines GPIO (Entradas/Salidas de propósito general), SPI, I²C, Interfaz I²S con DMA, pines dedicados a UART y un conversor ADC de 10-bit1. El chip puede ser programado con el entorno Arduino IDE o con otros lenguajes como Lua o MicroPython. Podemos apreciar en la siguiente figura cómo se distribuyen los pines de conexión y sus funciones.

III-B3. Esp-01: ESP-01 [12] es un módulo WiFi basado en el chip ESP8266 [13], permite a otros microcontroladores conectarse a una red inalámbrica Wi-Fi y realizar conexiones simples con TCP/IP usando comandos al estilo Hayes. El ESP-01 tiene 8 pines:

- 2 entradas/salidas de propósito general (GPIO0 y GPIO2).
- 2 para la alimentación (VCC y GND).
- 2 para la transmisión de datos (TX y RX)

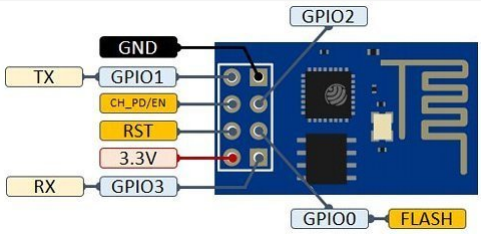
Figura 3. Placa ESP8266 en su variante 12-E



- 2 para el control del modo de funcionamiento (CH PD y RST).

Se destaca por su reducido tamaño, costo, consumo y la posibilidad de entrar en sueño profundo cuando no está siendo utilizado. En la siguiente figura puede observarse el dispositivo con un diagrama de sus conexiones.

Figura 4. Placa ESP01 con referencia de conexiones



III-B4. Protocolo MQTT: MQTT [14] es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue, que se basa en la pila TCP/IP para la comunicación. Fue creado en 1999 para su uso en la industria del petróleo y el gas, y se convirtió en un estándar en 2014. MQTT [15] funciona con un servicio de mensajería push con patrón publicador/suscriptor (pub-sub), donde los clientes se conectan con un servidor central llamado broker. Los clientes pueden publicar mensajes en topics organizados jerárquicamente, y otros clientes pueden suscribirse a esos topics para recibir los mensajes que les interesan. MQTT es un protocolo ligero y eficiente, que requiere pocos recursos y consume poco ancho de banda. Es ideal para la comunicación entre dispositivos de IoT, que suelen tener limitaciones de potencia, consumo y conectividad. MQTT también ofrece diferentes niveles de calidad de servicio y mecanismos de seguridad para garantizar la fiabilidad y la privacidad de los datos transmitidos.

III-B5. Multicast: Multicast [16] es una tecnología de red la cual permite enviar un mismo mensaje a varios dispositivos. Un mensaje multicast [17] es aquel que va dirigido a un grupo de dispositivos finales dentro de una misma subred o VLAN, para poder enviar este tipo de mensajes se debe utilizar una dirección de destino del tipo multicast. Para que los interesados reciban el mensaje, deben leerlo de la dirección multicast provista.

IV. Desarrollo

IV-A. *Repositorio*

Para mantener un orden de las plataformas a desarrollar se decide utilizar Github [18] como repositorio, de esta forma el proyecto estará disponible en un mismo lugar. El principal beneficio de utilizar un repositorio como Github es contar tanto con un versionado como con la posibilidad de manejar posibles colisiones que se producen al estar trabajando varias personas sobre un mismo archivo.

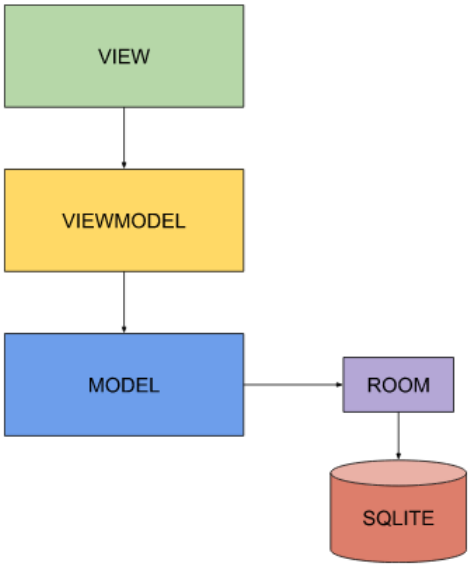
IV-B. Servidor

En principio vamos a establecer quien oficiará de servidor, ya que de esta forma luego podremos desarrollar los dispositivos correspondientes pudiendo probarlos con el mismo. Al comienzo se utilizó una Raspberry Pi [19] con el software Mosquitto [20] para oficiar de Broker, tras varias investigaciones se tomó la decisión de integrar el broker con una aplicación móvil que gestione los dispositivos que se conectan, de esta forma se les puede asociar un nombre al conectarse y conocer el estado de los mismos, esta misma aplicación será la encargada de derivar las notificaciones a los camareros según corresponda.

El aplicativo se decide hacer para la plataforma Android, ya que es la más predominante en Argentina y consta de una amplia gama de modelos a utilizar [21]. Como lenguaje se utilizará Kotlin, el cual es recomendado por Google [22] quien es el principal desarrollador de la plataforma y por la misma razón se utiliza como IDE Android Studio.

Para el desarrollo de la aplicación, se utiliza una estructura conocida como MVVM (Model-View-ViewModel) [23] la cual abstrae las capas de obtención de información y proceso de la misma de la vista, optimizando el funcionamiento. En la siguiente imagen puede verse un diagrama a modo de ejemplo.

Figura 5. Diagrama de modelo MVVM



Al tratarse de una prueba de concepto y para minimizar el esfuerzo se unifican los aplicativos de servidor y camareros, con un selector en el inicio que permita elegir el modo de uso del mismo, de esta forma se aprovecha el uso de las bibliotecas de conexión ejerciendo de broker o client dependiendo del modo seleccionado. En las siguientes imágenes se puede apreciar el selector y la pantalla de inicio del broker.

IV-B1. Conexiones: Al conectarse el dispositivo a una red ya establecida, el mismo no obtendrá siempre la misma dirección IP del DHCP, por lo cual se resuelve enviar un mensaje constante por multicast indicando la dirección actual del servidor, de esta forma todos los dispositivos que integren la red podrán leer el mensaje y conectarse al mismo. Para lograr esto se utiliza la biblioteca net incluida en Java, se crea una clase con todo lo necesario para el envío y la recepción de la ip por multicast de esta forma puede ser utilizada por el aplicativo tanto en modalidad broker cómo cliente. En cuanto a la conexión MQTT, tanto para broker como para cliente, se utiliza la biblioteca KMQTT Davide Pianca [24], esta integra las clases necesarias para establecer un broker y cliente MQTT en Kotlin, la misma puede ser utilizada con la tecnología Multi Platform de Kotlin por lo que a futuro será posible integrar a una App iOs.

Figura 6. Pantalla de selección principal

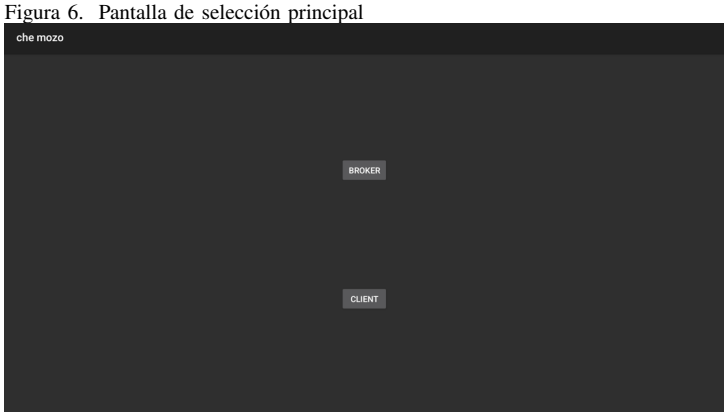


Figura 7. Pantalla de inicio en modo broker



IV-B2. Configuración: Para poder reconocer de forma sencilla los dispositivos conectados, se resuelve permitir configurar un alias, todos los dispositivos con conexión activa figuran en un listado, desde allí se podrá mantener pulsado para ingresar a un menú de opciones entre las que se encontrara la creación de un alias para el mismo. En la siguiente imagen se puede observar la opción de alias.

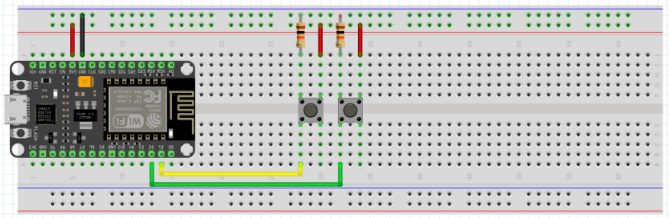
Figura 8. Opción de creación de alias



IV-C. Dispositivo en mesas

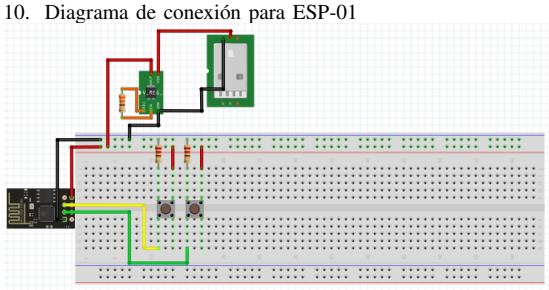
Para desarrollar el dispositivo que se encontrará en las mesas y será utilizado por los comensales se emplea una placa de desarrollo basada en esp8266, con un adicional de dos pulsadores y dos resistencias. Para este prototipo contaremos con la placa más conocida como ESP8266 12-E la cual nos dispone un set de entradas y salidas listas para programar las cuales podremos integrar con los pulsadores y así obtener la interacción del comensal, además ofrece alimentarlo y programarlo utilizando un cable usb lo que facilitará el conexionado para las pruebas. En la siguiente figura se puede apreciar cómo se conectan los pulsadores con la placa de desarrollo, las resistencias de 100k ohm son utilizadas para evitar corrientes parásitas que provoquen una falsa respuesta de pulsación.

Figura 9. Diagrama de conexión para ESP8266 12-E



Como ejemplo se deja el conexionado para ESP-01 con su correspondiente fuente de energía y conversor de tensiones en la siguiente figura.

Figura 10. Diagrama de conexión para ESP-01



El desarrollo se realiza en el lenguaje C, ya que es el más utilizado y nos permite utilizar el software Arduino IDE. Para lograr utilizar el módulo WIFI se requerirá la biblioteca ESP8266WIFI [25], para mantener la conexión con el servidor y recibir paquetes por multicast utilizaremos la biblioteca WIFIUdp [26] y finalmente para lograr la conexión con el servidor MQTT haremos uso de la biblioteca PubSubClient [27]. En cuanto a la estructura nuestro código se divide en dos secciones, setup la cual es donde tendremos todo lo referente a configuración y conexiones, aquí se conectaba al wifi (el usuario y contraseña debe estar programado en el dispositivo) y quedará a la espera de recibir por multicast la IP donde se encuentra el servidor MQTT, una vez la obtiene procede a realizar la conexión y de ser exitosa pasará a la segunda sección, la cual se denomina loop. Es en la sección loop donde se monitorea la pulsación de los dos botones disponibles y en caso de detectar que fueron pulsados se enviará el mensaje correspondiente a nuestro servidor MQTT.

En la siguiente figura se puede apreciar una porción de nuestro código loop desde Arduino IDE.

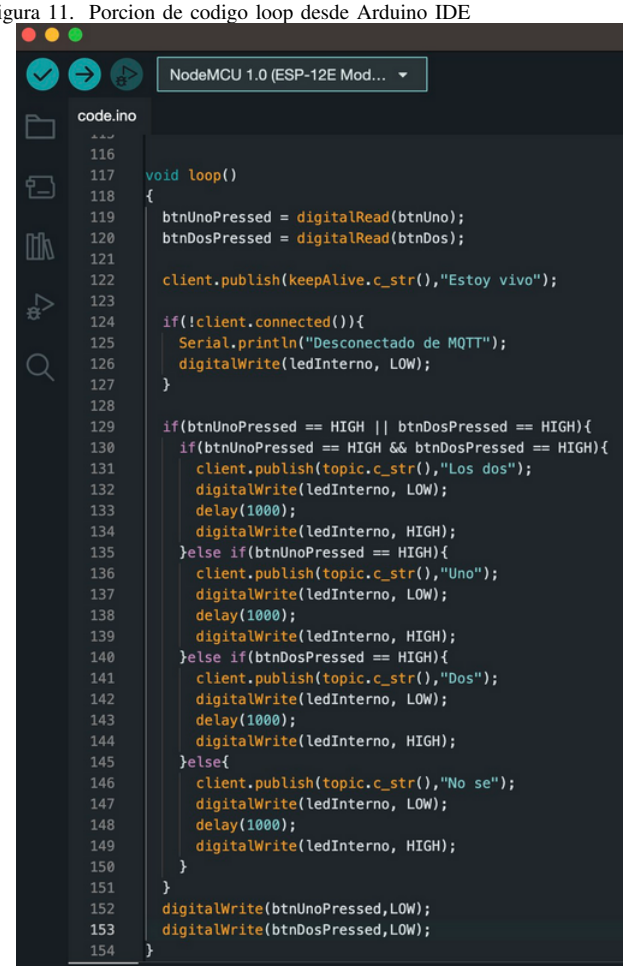
Para poder detectar problemas de conexión sin necesidad de estar conectado a un ordenador, se dispuso de un código de luz con las siguientes características:

- Parpadeo rápido = Conectando a la red.
- Led encendido = Esperando IP de servidor.
- Led encendido al pulsar = Pulsación detectada..
- Led encendido = Se desconectó del servidor.

IV-D. Dispositivo para camareros

Al tomar la decisión de utilizar la misma aplicación tanto para el servidor como para los camareros, a este punto gran parte del desarrollo está realizado, resta resolver cómo llegan los mensajes provenientes del servidor y cómo se hará para lograr que sean recibidos aun en un salón lleno de personas. Para los mensajes, por el momento se reutilizará el protocolo MQTT ya implementado para el servidor, se creará un nuevo tema al que los aplicativos de camareros puedan suscribirse y recibir los mensajes. Tras la recepción de un mensaje, el dispositivo emitirá un alerta sonoro y vibrará para llamar la atención del usuario, luego en pantalla mostrará de dónde proviene el mensaje para indicarle al camarero desde donde es solicitado y el motivo. Como solución a la posibilidad de que, al ser un ambiente ruidoso, no pueda escuchar la alerta, se propone la utilización del

Figura 11. Porcion de codigo loop desde Arduino IDE



sistema de notificaciones en Android. Al hacer uso de este sistema puede incluirse un reloj smart el cual canalice las notificaciones del dispositivo móvil. Este al estar en la muñeca del usuario hará más sencilla la tarea de notificarlo.

V. Conclusión

Haciendo un balance de las soluciones logradas se concluye que el propósito fue logrado, se logró una alternativa para la comunicación en los locales gastronómicos y, en el transcurso, se pudo investigar y lograr la implementación y creación de dispositivos IOT, aplicaciones y protocolos. Entre las situaciones que no estaban planificadas al comenzar la investigación se destacan la utilización de la aplicación como broker, la posibilidad de utilizar multicast para la obtener la ip del servidor y la implementación de notificaciones para smartwatch. Estas soluciones son gracias a personas y grupos que generan desarrollos y no solo los hacen públicos sino que también permiten su utilización sin coste alguno, lo cual permite no solo reducir tiempos y trabajo sino también poder experimentar con los mismos y ver la viabilidad de su utilización. Quedan cosas por hacer y mejorar en este proyecto, como también existen posibilidades de expansión, algunas de ellas quedarán planteadas en la sección a continuación, las demás será producto de la imaginación de quienes vean este proyecto y decidan continuarlo.

VI. Líneas futuras de investigación

VI-A. Aplicación multiplataforma

El proyecto de la aplicación móvil se deja planteado utilizando Kotlin Multiplatform [28], esta tecnologia permite abstraer la capa

lógica de la visual, permitiendo que la capa lógica sea utilizable en varias plataformas como iOs, Windows, MacOs o Linux mientras que puede configurarse una capa visual independiente por sistema realizada en el lenguaje de preferencia cada uno.

VI-B. Websocket y API para Cliente/Servidor

Se considera que la utilización del protocolo MQTT entre aplicaciones móviles no es el ideal, al canalizarse la comunicación de los dispositivos en el servidor y que luego sea él mismo quien se comunique con los clientes, se deja la puerta abierta a la utilización de otros protocolos y tecnologías para lograr que sea más rápida, eficiente y genere menos consumo, como por ejemplo, la utilización de una api con websocket donde se conecten los clientes para mantener escucha activa.

VI-C. Llamados a camareros desde otras entidades

Los llamados a mozos están limitados a la utilización de los dispositivos, pero esto puede ampliarse a por ejemplo llamados desde web o aplicaciones que se integren directamente al servidor y luego las canalice a los clientes utilizados por los camareros.

REFERENCIAS

[1] J. Morgan, “A Simple Explanation Of ’The Internet Of Things’”, May, 2014. Forbes
[2] Paakat: Revista de Tecnología y Sociedad, núm. 4, octubre, 2012. redalyc.org
[3] Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T. M. T., ... Miettinen, M. (2013). From big smartphone data to worldwide research: The mobile data challenge. Pervasive and Mobile Computing, 9(6), 752-771. Idiap
[4] Kodali, R. K., Mahesh, K. S. (2016, December). Low cost ambient monitoring using ESP8266. In 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I) (pp. 779-782). IEEE. IEEE
[5] Soni, D., Makwana, A. (2017, April). A survey on mqtt: a protocol of internet of things (iot). In International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017) (Vol. 20, pp. 173-177). Articulo
[6] J.Teicher, “The little-known story of the first IoT device”, Feb, 2018. IBM
[7] International Conference for Industry and Academia, Mar, 2008 Iot conference
[8] A. Gabbai, “Kevin Ashton Describes the Internet of Things”, Ene, 2015. Smithsonian Magazine
[9] Google Trends, búsqueda de la palabra clave “IoT”, desde el 01/01/2018 al 01/01/2023. Google Trends
[10] Arduino DataSheet
[11] Espressif DataSheet
[12] Microchip, Esp01 DataSheet
[13] International Journal of Electronics and Communication Engineering and Technology(IJECET) Volume 6, Issue 8, Aug 2015 Articulo
[14] MQTT, Mosquitto Articulo
[15] MQTT, Oasis Articulo
[16] Sylvia Ratnasamy, Andrey Ermolinskiy, Scott Shenker, Revisiting IP Multicast, Nov, 2006 Articulo
[17] Katia Obraczka, Multicast Transport Protocols: A Survey and Taxonomy, Nov, 1997 Articulo
[18] Zagalsky, A., Feliciano, J., Storey, M. A., Zhao, Y., Wang, W. (2015, February). The emergence of github as a collaborative platform for education. In Proceedings of the 18th ACM conference on computer supported cooperative work social computing (pp. 1906-1917). Articulo
[19] Raspberry Pi, DataSheet Articulo
[20] Mosquitto, DataSheet Articulo
[21] Marcelo Bellucci, Las 10 marcas de celulares más vendidas del mundo: quién gana la guerra de los Android, Feb, 2023 Articulo
[22] Android developer Articulo
[23] Lou, T. (2016). A comparison of Android native app architecture MVC, MVP and MVVM. Eindhoven University of Technology. Articulo
[24] Davide Pianca, KMQTT GitHub
[25] ESP8266WIFI Esp8266 Documentation
[26] ESP8266 Community Forum, WiFiUdp GitHub
[27] Nick O’Leary, pubsubclient GitHub
[28] kotlinlang, Kotlin Multiplatform Articulo

VII. CV



Diego Ginko nació un 21 de diciembre de 1991 en la Capital Federal, Buenos Aires, Argentina. En 2009 se recibe como Tecnico en electronica de la EET N°9 de Lanús Ingresa a trabajar como técnico de soporte a la empresa Maxirest en 2013, allí se dedica a dar asistencia a locales gastronómicos tanto por vía telefónica como presencial. En marzo de 2016, luego de iniciar la tecnicatura en desarrollo de software pasa a formar parte del área de desarrollo, allí comienza a analizar y generar soluciones para la gastronomía. A partir de 2018 se dedica íntegramente al desarrollo de aplicaciones móviles, desde el relevo de la necesidad, diseño de arquitectura, programación y mantenimiento. En 2018 se recibió como técnico superior en desarrollo de software, en ISTEA y está finalizando la carrera para convertirse en Licenciado en sistemas de la información en la Universidad de Palermo.