Comprensión del Lenguaje Natural e Integración de Asistente Virtual para el manejo de cuentas y comercios

Wilfredo Sierra Rivera
Maestría en Ingeniería de Computadoras
Nelliud Torres, DBA
Departamento de Ingeniería
Universidad Politécnica de Puerto Rico

Resumen *— En Puerto Rico, las compañías no han explorado la posibilidad de incorporar el uso de un Asistente Virtual para manejar transacciones comerciales. En búsqueda de facilitar la interacción de un usuario con una aplicación sin la necesidad del uso directo de un móvil o computadora, se creó una solución integrando un Asistente Virtual. Para esto se llevó a cabo una compresión del lenguaje natural y se utilizaron varias herramientas como Amazon Web Services, Alexa y NodeJS, entre otros. Dicha solución nos permite solicitarle al Asistente Virtual realizar pagos de servicios como agua, luz y telefonía, entre otros. Para ello, se integró una base de datos en MySQL para almacenar los datos de la aplicación web. También se añadió el* framework *Express para establecer la comunicación entre la web y la aplicación. Además, se utilizaron funciones Lambda para poder almacenar los datos de las transacciones en la base de datos del* cloud *y un API Gateway para la comunicación entre la aplicación web y el* cloud*. Este proyecto es una muestra pequeña de todo lo que se puede lograr integrando esta solución a otras aplicaciones.*

Términos clave *— asistente virtual, comandos de voz, comprensión de lenguaje natural, servicios inteligentes de voz*

INTRODUCCIÓN

Los servicios inteligentes de voz y lenguaje natural se han vuelto una herramienta útil para complementar el uso de dispositivos inteligentes. Compañías como Amazon, Google y Apple tienen sus propias versiones que utilizan comandos de voz para simular conversaciones y realizar tareas específicas. Alexa, Google Assistant y Siri son productos conocidos y usados por muchos, pero ¿hasta qué punto podemos extender su utilidad y cuán variadas y extensas son sus opciones? Tomando esto en cuenta, se creó una solución para integrar estos servicios de Asistente Virtual y comprensión del lenguaje natural para el manejo de pagos y cobros en cuentas personales y comercios.

DEFINICIÓN DE TÉRMINOS

- **Alexa Skill:** funcionalidades extras que se pueden añadir al Asistente Virtual.
- **Amazon Resource Names (ARN):** identifican de forma exclusiva los recursos de Amazon Web Services.
- **Application Programming Interface (API):** conjunto de definiciones y protocolos para ser utilizados por otro software.
- **API Gateway:** servicio que facilita la creación, la publicación, el mantenimiento, el monitoreo y la protección del API.
- **Amazon Web Services (AWS):** colección de servicios de computación en la nube pública que en conjunto forman una plataforma de computación en la nube.
- ***Back end*:** capa de acceso a los datos que no es directamente accesible a los usuarios y que contiene la lógica de manejo de estos.
- **Bootstrap4:** *framework* de CSS, HTML y JavaScript para desarrollar webs que se ajusten a cualquier resolución y dispositivo.
- ***Cron job*:** método para manejar tareas repetitivas.
- **CSS:** lenguaje para darle estilo a un documento de HTML.
- **DynamoDB***:* servicio de base de datos NoSQL ofrecido por Amazon.
- ***Endpoint*:** URL que reciben o retornan información de una web.
- **Entity Relationship Diagram (ERD):** modelo para representar entidades de una base de datos.

- **Entorno de Desarrollo Integrado (IDE): s**istema para el diseño de aplicaciones que cuenta con un editor de código, un compilador y un depurador en una misma interfaz gráfica.
- **Express:** *framework* de aplicación web de *back end* para NodeJs.
- ***Front end*:** parte de un programa con las tecnologías de diseño y desarrollo que se encargan de la interactividad con los usuarios.
- **HTML:** lenguaje de marcado para elaboración de páginas web.
- ***Intents*:** acciones que se le permiten realizar al usuario en el asistente virtual.
- **JavaScript:** lenguaje de programación interpretado para la creación de páginas web dinámicas.
- **Lambda:** servicio que permite ejecutar códigos sin un servidor físico.
- **MySQL:** sistema de gestión de bases de datos relacionales.
- **Navbar:** sección de una interfaz gráfica para acceder a información.
- **NodeJs:** entorno que permite ejecutar JavaScript en el servidor.
- ***Slots*:** variables que el usuario incluirá en sus expresiones al comunicarse con el asistente virtual.
- ***Utterances*:** diferentes expresiones que podrá utilizar el usuario con el asistente virtual.
- **XMLHttpRequest:** interfaz para realizar peticiones HTTP y HTTPS a servidores web.

PROBLEMA

Actualmente en Puerto Rico, las compañías públicas y privadas se han esforzado en desarrollar aplicaciones web y móviles con procesos manuales y automatizados. Sin embargo, han ignorado un nuevo método que podría facilitar la interacción entre usuarios y productos. Este método se conoce como los Asistentes Virtuales tales como Alexa, Google Assistant y Siri. La idea es integrar en las compañías esta tecnología como parte de una solución para facilitar, aumentar el interés y mejorar la experiencia de los usuarios.

METODOLOGÍA

Se creó un proyecto con tres partes: Aplicación Web, *Cloud* y Asistente Virtual. Para comenzar con la primera parte, se identificaron las herramientas, el lenguaje de programación, la base de datos y el proveedor de servicios de *cloud*. A base del alcance al que se quiere llevar este proyecto, se seleccionó NodeJS como principal lenguaje de programación, debido a su compatibilidad con el *cloud* y porque es un lenguaje que nace de JavaScript. JavaScript es uno de los lenguajes más utilizados en el mundo y puede servir para darle acción a una página web, validar en *client-side* y para *back end*.

La base de datos que se seleccionó fue MySql y el *framework* que se utilizó fue Express para facilitar la comunicación entre NodeJS, MySql y la Web [1], [2]. El IDE que se utilizó fue Visual Studio, debido a que ya tiene integrado Express para la creación de un proyecto de NodeJS.

Se utilizó Amazon Web Services (AWS) como el *cloud* para el proyecto, debido a la alta demanda de servicios útiles que están integrados [3]. Además, AWS cuenta con la ventaja de que se le puede configurar un API Gateway para que tanto la aplicación web como el Asistente Virtual, que en este caso fue Alexa, pueda utilizar los servicios como mediador.

Compresión del lenguaje natural

Con la comprensión del lenguaje natural, las computadoras pueden deducir lo que un hablante dice [4]. En otras palabras, permite que la tecnología de voz prediga lo que se le está pidiendo. Las tecnologías de voz reconocen patrones y significados dentro del lenguaje humano, lo que le permite a la computadora entender lo que se está diciendo sin la necesidad de preguntarlo de una manera específica y haciendo parecer que se está llevando a cabo una conversación real.

Alexa

Se utilizó como Asistente Virtual a Alexa para no tan solo hacer las acciones por medio de una plataforma web o móvil, sino para expandir la

experiencia de un usuario que no tiene un dispositivo a la mano. Para esto, se analizó la manera en que opera Alexa con una solución personalizada. El desarrollo se trabajó desde la consola de Alexa en la página oficial de desarrollo de Amazon. Luego de acceder a dicha consola se autenticó con la cuenta personal, se procedió creando un Alexa Skill, asignándole el nombre que va a identificar la solución personalizada y seleccionando lenguaje de programación NodeJs.

Una vez establecido, se configuraron los componentes *intents*, *utterances* y *slots* para que cuando un usuario pregunte o solicite algo a Alexa, esta pueda contestarle. El primer paso fue la creación de los *intents* en la página principal de la consola. Los *intents* son las acciones que se realizan una vez el usuario solicita o dice algo a través de Alexa. En otras palabras, son similares a los controladores. Tan pronto se carga la pantalla dentro de *intent*, aparece una cajita donde se agregó una lista de oraciones o frases. Cuando el usuario hable a través de Alexa, el *intent* recibe la petición e internamente valida si la oración o las frases que solicitó el usuario están dentro de la lista creada. Esta lista de oraciones y/o frases es lo que se le conoce como *utterances.*

No obstante, dentro de las oraciones o frases se agregó unas palabras entre los símbolos {}. Por ejemplo: "Quiero comprar un carro con la marca **{marca}** y el modelo **{modelo}**". Estas palabras que están entre los símbolos representan lo que se conoce como *slots.* Los *slots* son llaves o variables que contienen el valor cuando el usuario solicita algo a través de Alexa. Por ejemplo: "Quiero comprar un carro con la marca **Toyota** y el modelo **Corolla**".

Una vez se completó la configuración, se presionó en la consola una opción llamada "Code" para cargar un documento en el lenguaje JavaScript, llamado "index.js". Esto tiene predeterminado una constante llamada "LaunchRequestHandler", que por defecto es el primer *intent* que es llamado. En otras palabras, cuando se dice: "Alexa, open My Accounts", lo primero que responderá Alexa está en esa constante bajo el método "handle".

Finalmente, se agregó una constante, y dentro de esta se agregaron dos métodos "canHandle" y "handle". En el método "canHandle", se estableció la conexión con el nombre del *intent* creado. Cuando el usuario solicite algo por medio de Alexa, la oración o frase se corrobora internamente en la lista de *uterrances* y *Alexa* va a invocar el método "handle". Este método recogerá los valores *slots* dentro de la oración o frase que diga el usuario. La página oficial de desarrollo de Amazon cuenta con una herramienta para probar la conversación entre el usuario y Alexa.

Cloud

Para el *cloud* AWS, se comenzó generando una política para establecer los permisos, se creó un rol y se relacionó dicho rol con esa política. Una vez establecido, se creó la base de datos en DynamoDB para poder obtener e insertar los registros que vienen por medio de la aplicación web y el Asistente Virtual [5].

Integración entre el *cloud* y la aplicación web

Para poder integrar lo que está en la aplicación web al *cloud* AWS, se generó una función en el lenguaje JavaScript que contiene un objeto de tipo XMLHttpRequest para poder hacer un llamado por el protocolo HTTPS al API Gateway de AWS y poder hacer uso de la función Lambda [6], [7].

Integración entre el *cloud* y Alexa

Existen varias maneras para conectar Alexa con AWS, pero la más popular es creando una función Lambda dentro de AWS. Tan pronto se creó la nueva función Lambda en AWS con el lenguaje de programación NodeJS, se copió el Amazon Resource Names (ARN) de la función Lambda creada. Luego se accedió a la página oficial de desarrollo de Amazon y se seleccionó la opción "Endpoint". Por último, en el campo de "Default Region", se colocó el ARN y se guardó la información.

Una vez se accedió a la página oficial de desarrollo de Amazon, se copió el *skill id* y se colocó en el campo correspondiente la página de "Manejo de AWS". De esa manera se creó la comunicación entre AWS y Alexa (figura 1) [8].

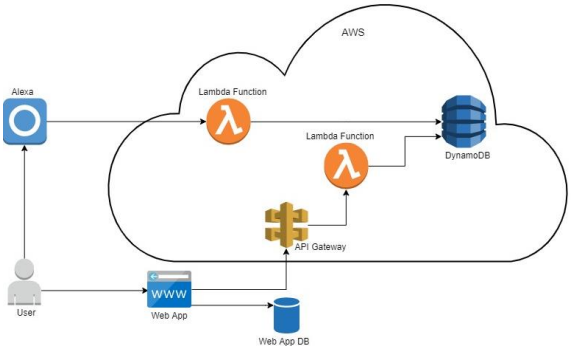

Figura 1

Diagrama conceptual del proyecto

Aplicación web

Para esta aplicación se diseñó un Entity Relationship Diagram (ERD) con todas las tablas y relaciones que se utilizaron para la creación de la base de datos en MySql. El énfasis principal de esta base de datos son las entidades, los proveedores de servicios y las cuentas. Luego, se creó un proyecto en Visual Studio utilizando como lenguaje de programación NodeJS. Se descargaron aquellas librerías útiles para el proyecto y se estableció la configuración entre el *framework* y la base de datos de MySql. Una vez se completó la configuración, se crearon dos carpetas: una llamada "routes" y otra llamada "views" (figura 2). En la carpeta "views" están las plantillas de *front end* que, en vez de tener la extensión .html, tendrán la extensión .pug. Esto es básicamente lo mismo; la única diferencia consiste en que en los archivos con extensión .pug es vital la indentación, y que además en estos se puede reutilizar código para evitar la duplicidad integrando partes de otros archivos con extensión .pug y validaciones dentro del *front end* [9].

Por su parte, en la carpeta de "routes" (figura 2) están todos los documentos de JavaScript con el código de *back end.* En este último es donde se encuentra la lógica del *server-side* (la acción tiene lugar en el servidor web) y las consultas a la base de datos de manera asíncrona (los cambios no se hacen al momento, tardan un tiempo en completarse).

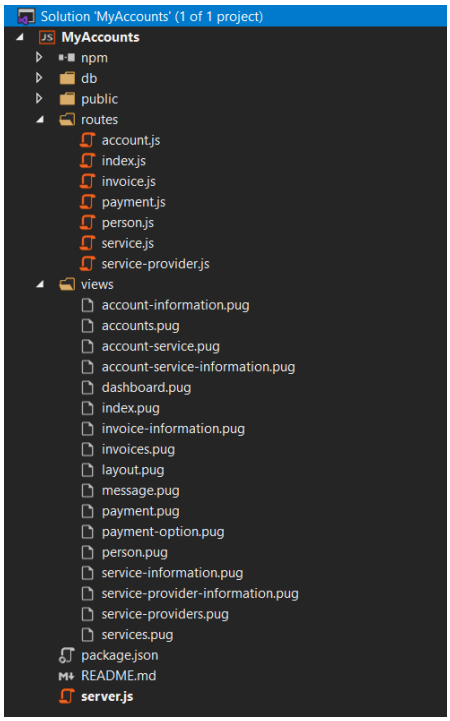

Figura 2

Estructura de la aplicación web, parte I

El *front end* está hecho con Bootstrap4 y entre las librerías que más se utilizaron se encuentran:

- **JQuery con funciones Ajax:** por medio de alguna acción en *front end* hace un llamado al *back end* sin tener que cargar la pantalla y evita perder la información entrada por el usuario [10].
- **Express-Validator:** para validar los campos que se envían del *front end* al *back end*. Por ejemplo, los campos requeridos y campos para los que el *back end* solo espera números.
- **Express-Messages:** para mostrar un mensaje al *front end*.
- **Dateformat:** para el formato de fecha en el *back end*.
- **MySQL:** uso de llamadas para conexión a la base de datos.

Asimismo, en la carpeta "db" (figura 3) se encuentran los valores para la conexión de la base de datos. Por su parte, en la carpeta "public" (figura 3) se encuentra todo el contenido de CSS y JavaScript para el uso de la página web. Por último, el archivo "server.js" contiene la configuración entre la aplicación, el *framework* y el servidor local.

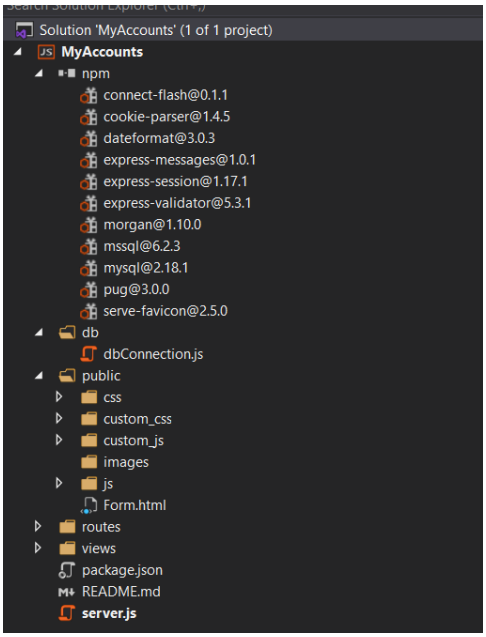

Figura 3

Estructura de la aplicación web, parte II

RESULTADOS

A base de lo establecido en la metodología, se creó un proyecto en el cual se pusieron a prueba los conceptos anteriores mediante la Comprensión del Lenguaje Natural e Integración de Asistente Virtual, particularmente en el manejo de cuentas y comercios. El nombre de este proyecto es "My Accounts".

En esta aplicación, el usuario puede realizar pagos y agregar y editar cuentas de servicios corrientes como el agua, la luz y el cable, entre otros. El enfoque principal de este proyecto es llevar a cabo estas acciones por medio de un Asistente Virtual más allá de poder hacerlo mediante la aplicación web.

Para el Diagrama de Entidad Relacional de la Base de Datos (figura 4) se usan las siguientes entidades:

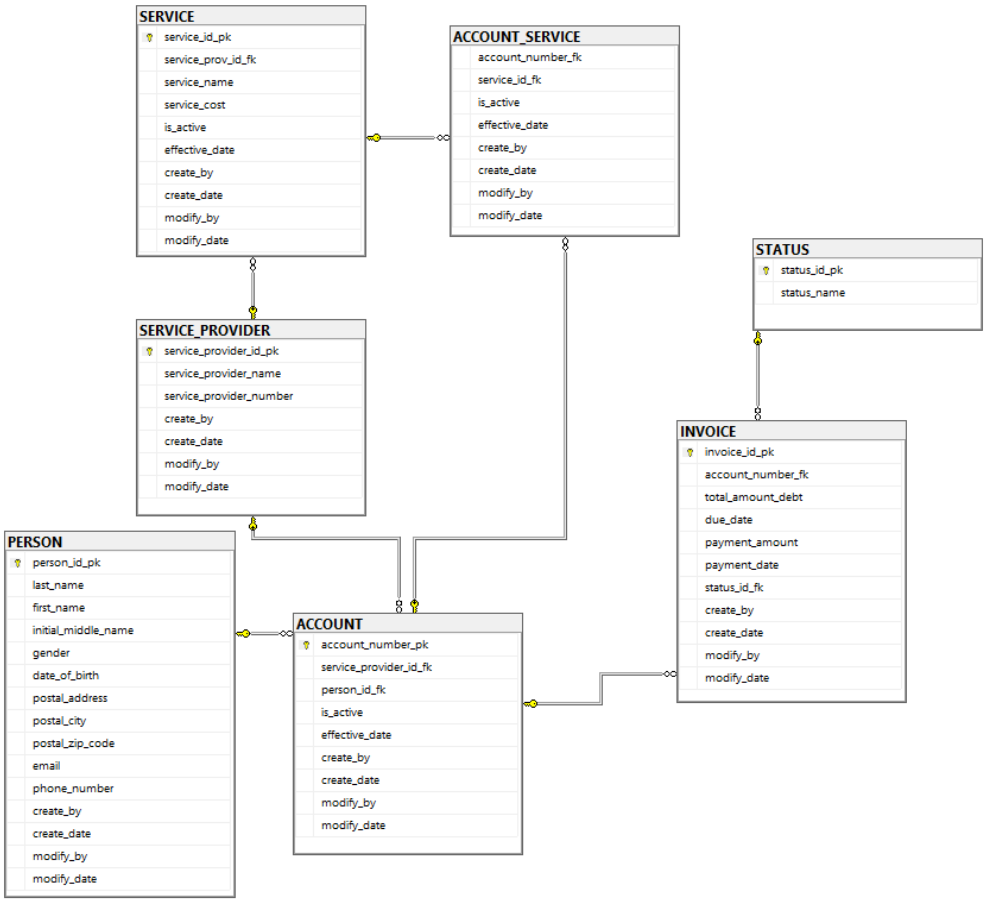

Figura 4

Diagrama de Entidad Relacional de la Base de Datos

- **PERSON:** aquí se encuentran los datos que incluyen identificador de persona, nombre, inicial, apellido, género, fecha de nacimiento, dirección postal, ciudad, código de área, correo electrónico y número de teléfono.
- **ACCOUNT:** tiene una relación de muchos a muchos entre las tablas "PERSON" y "SERVICE_PROVIDER". Se encuentran los datos del identificador de persona, el identificador de proveedor de servicio, el número de cuenta, si la cuenta está activa o no, y el día cuando fue efectiva la cuenta.
- **SERVICE_PROVIDER:** en esta se encuentran los datos del nombre del proveedor de servicio y el identificador del proveedor de servicio.
- **SERVICE:** tiene una relación de uno a muchos con la tabla "SERVICE_PROVIDER". Se encuentran los datos del nombre del servicio, el costo del servicio, el identificador del servicio, si el servicio está activo o no, y el día cuando el servicio fue efectivo.
- **ACCOUNT_SERVICE:** tiene una relación de muchos a muchos entre las tablas "SERVICE" y "ACCOUNT". Se encuentran los datos del identificador del servicio, el número de cuenta, si esta activo o no el registro, y el día cuando fue efectivo el registro.
- **INVOICE:** tiene una relación de uno a muchos con la tabla "ACCOUNT". Se encuentran los datos del identificador de la factura, el número de cuenta, el monto total a pagar, la fecha de vencimiento del pago, el pago que realizó el usuario, la fecha de emisión de pago y el estatus de la factura.
- **STATUS:** tiene los datos del nombre del estatus de la factura y el identificador del estatus. A continuación, la lista de los estatus:
    - **Sent:** factura enviada a la persona.
    - **Pending Payment:** factura pagada por la persona, pero aún no se ha procesado por completo.
    - **Paid:** factura pagada.
    - **Overdue:** factura ha pasado del día de vencimiento de pago.
    - **Void:** factura revocada.

Todas estas tablas, excepto "STATUS", cuentan con las siguientes columnas para propósito de auditoría: "created_by", "created_date", "modified_by" y "modified_date".

Por el momento, este proyecto, dentro del aplicativo web, cuenta con las funciones de agregar una cuenta existente bajo un proveedor de servicio, inactivar alguna cuenta y realizar pagos. Por su parte, el asistente virtual solo cuenta con la función de realizar pagos. Como parte del desarrollo y para evitar ejecutar consultas manuales, se agregaron seis funciones: crear una factura, modificar una factura, añadir un proveedor de servicios, inactivar un proveedor de servicios, crear servicios y modificar servicios. Estas funciones no son públicas, ya que necesitan una interfaz en la cual el usuario se autentique y se identifiquen sus roles, permisos y accesos.

En la página de inicio se despliegan dos tablas (figura 5). La tabla superior contiene la información de los pagos pendientes y la tabla inferior contiene los balances de las cuentas. También contiene una pestaña con una tabla para el historial de pagos. En el "navbar" hay una opción llamada "Accounts", que redirige a una página donde aparecerán las cuentas. Es ahí donde se pueden editar o añadir nuevas cuentas (figura 6).

En la página de "Añadir cuentas", se encuentra un *dropdown* donde se puede seleccionar el proveedor de servicio, y hay un campo donde se coloca el número de cuenta asociado a este proveedor (figura 7). Una vez completada esta información, se guardará en la base de datos de la aplicación. En el caso de "Editar", el único campo que se puede modificar es el de activar o desactivar la cuenta (figura 8).

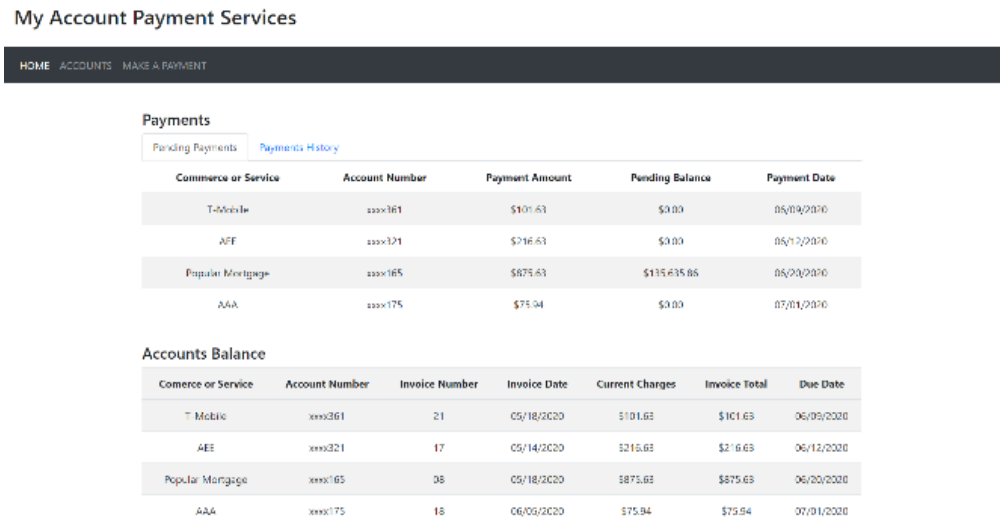

Figura 5

Página de inicio de la aplicación web

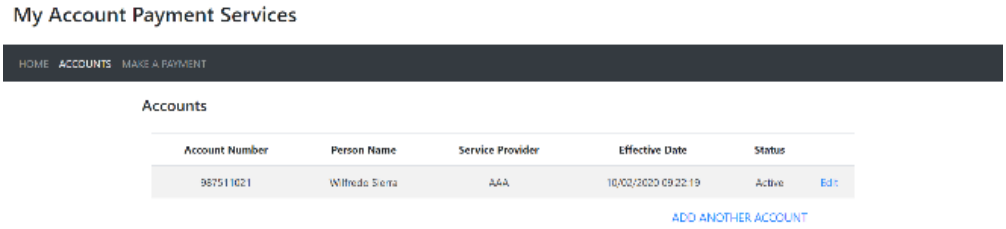

Figura 6

Página de "Accounts"

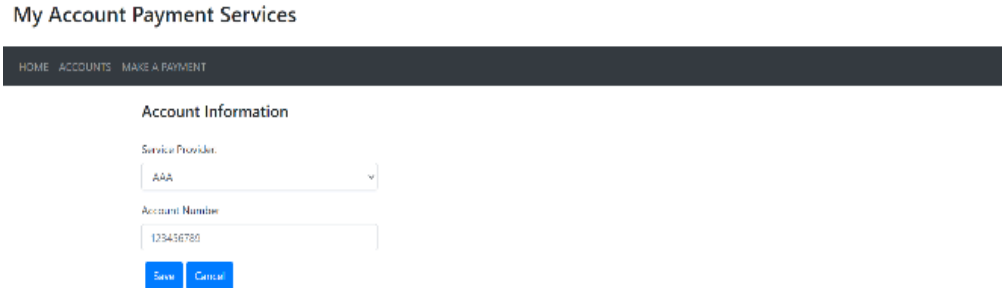

Figura 7

Página para añadir una cuenta

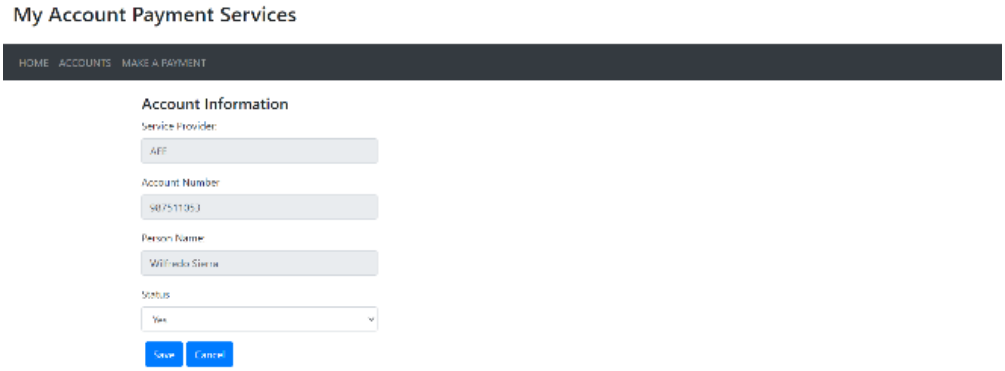

Figura 8

Página para Editar las Cuentas

En el “navbar” también hay una opción llamada “Make a Payment”. En esta página, al seleccionar la cuenta que se desea pagar, el sistema va a poblar los campos “Service Provider” y “Payment Amount”. Este último se llena en base al monto que se refleja en la factura y permite modificar la cantidad para pagar más o menos del monto establecido. Asimismo, cuenta con una validación que solo permite ingresar números (figura 9).

Luego de emitir el pago, el sistema envía los valores invocando el API Gateway de AWS que está atado con la función Lambda, y esta se encarga de insertar los valores del pago a la tabla “Transaction” en DynamoDB (figura 10).

Al establecer la comunicación entre Alexa y Lambda Function, se probó en la consola de Alexa el abrir la aplicación, emitir un pago y guardarlo en la base de datos de DynamoDB (figuras 11 y 12).

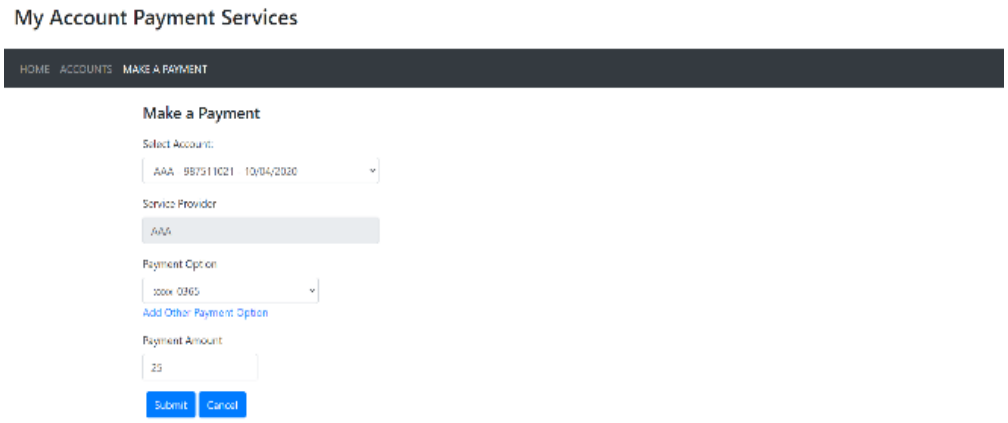

Figura 9
Página para Realizar Pagos

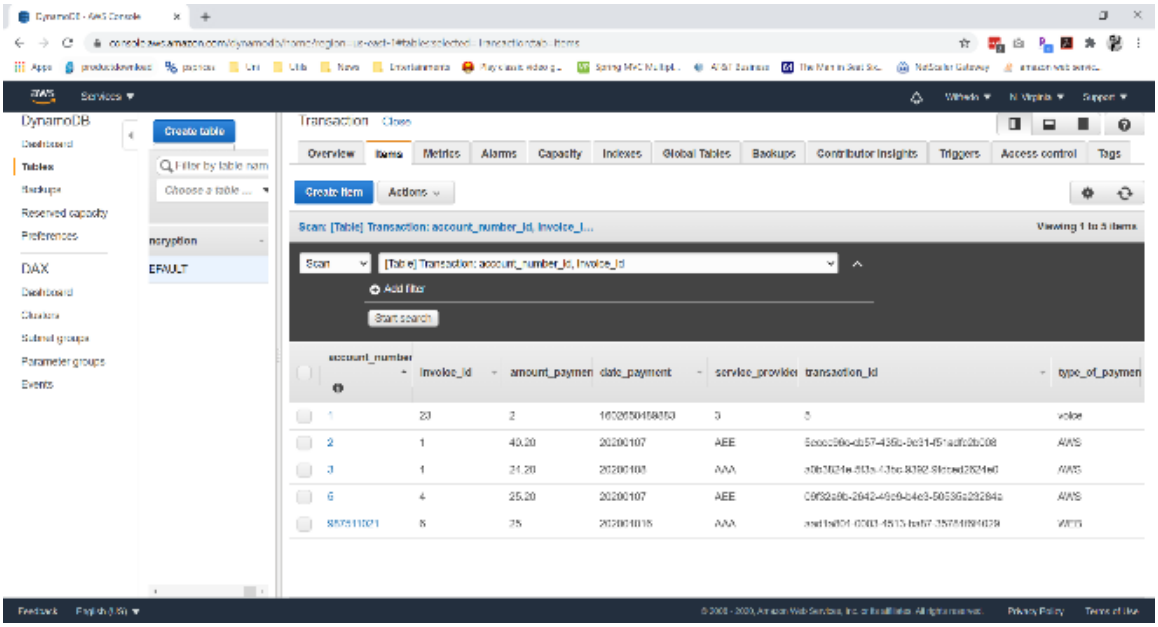

Figura 10
DynamoDB (AWS)

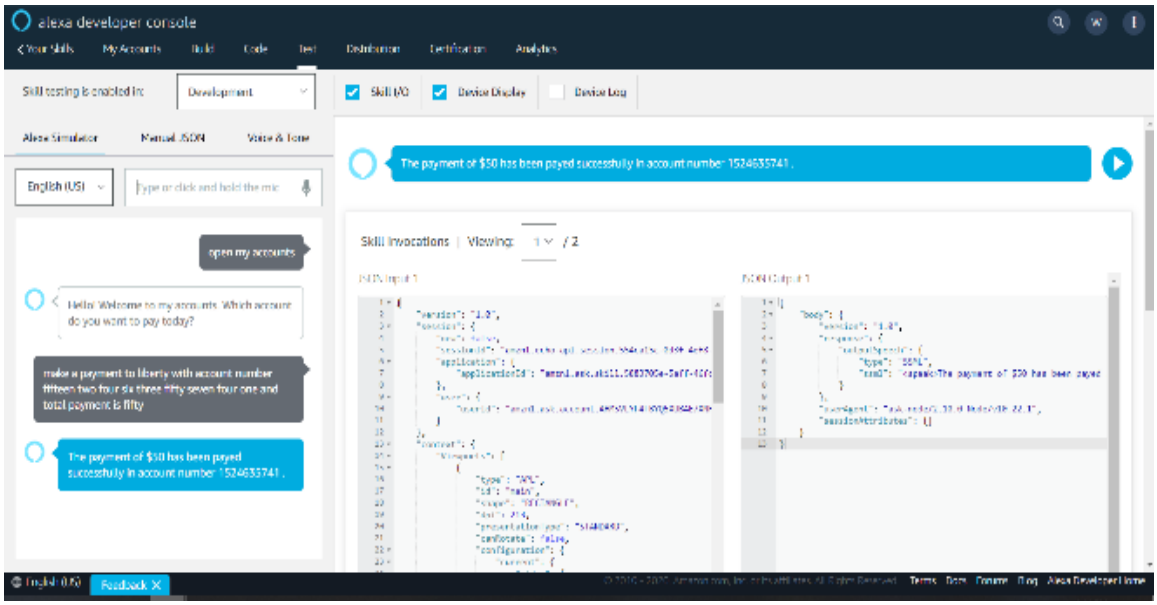

Figura 11

Conversación a través de Alexa

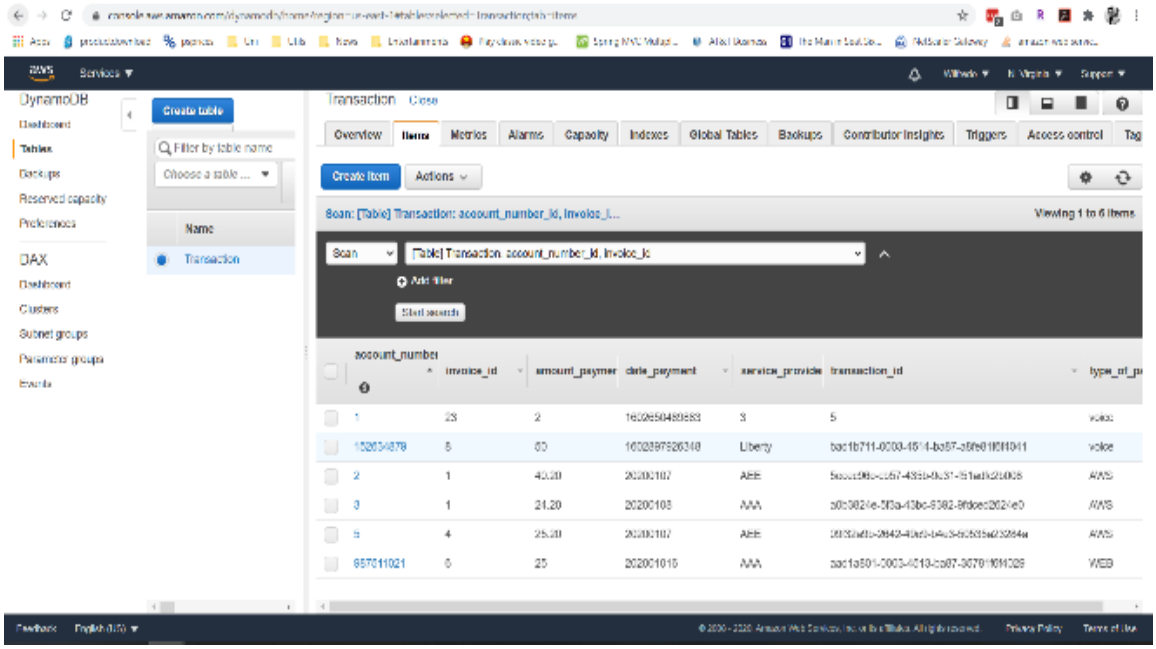

Figura 12

Resultado de DynamoDB por medio de Alexa

CONCLUSIÓN

La aplicación de “My Accounts” busca integrar nuevas tecnologías mediante la Comprensión de Lenguaje Natural y Asistente Virtual Alexa como una alternativa para facilitar la gestión de pagos de una manera dinámica y práctica. Este proyecto prueba que las posibilidades para integrar estas tecnologías en soluciones del día a día no solo son posibles, sino que cuentan con amplias posibilidades. Esta solución fácilmente se podría proponer como una opción adicional de pago para los comercios.

Aun así, son muchos los detalles que se pueden incluir en esta aplicación. Por ejemplo:

- Integrar un *cron job* que se ejecute a diario para poder obtener los valores de las transacciones que fueron procesadas y logre actualizar la información de las facturas.
- Agregar diálogo adicional entre Alexa y el usuario.
- Crear una página web donde el usuario se pueda autenticar.
- Crear tablas en la base de datos donde se almacenen las credenciales, los roles y los permisos de los usuarios.
- Exponer esta solución como una alternativa de pago a aquellos proveedores de servicios que no cuenten con una plataforma para generar facturas y pagos.
- Proponer esta solución a agencias de gobierno o privadas, para automatizar los procesos de cobro, el manejo de reclamaciones y la atención al cliente.

De igual modo, hay compañías como Evertec que tienen productos como ATH Móvil y Mi Banco a los que fácilmente se les puede integrar un asistente virtual. Este es el mismo caso para la Autoridad de Energía Eléctrica, el CRIM, Auto Expreso y cualquier otra compañía que cuente con una aplicación web para realizar pagos. Cabe mencionar que en Puerto Rico la única compañía que ha implementado una solución parecida es Seguros Múltiples, aunque esta solo funciona para el área de asistencia al cliente y no para pagos.

REFERENCIAS

[1] Express, “Application”. Accedido el 26 de agosto de 2020. [En línea]. Disponible: https://expressjs.com/en/4x/api.html#app

[2] Express, “Database integration”. Accedido el 26 de agosto de 2020. [En línea]. Disponible: https://expressjs.com/en/guide/database-integration.html

[3] Amazon Web Services, “Acerca de AWS”. Accedido el 10 de septiembre de 2020. [En línea]. Disponible: https://aws.amazon.com/es/about-aws/

[4] Amazon Alexa, “Alexa Skills Kit”. Accedido el 15 de septiembre de 2020. [En línea]. Disponible: https://developer.amazon.com/en-US/alexa/alexa-skills-kit/nlu

[5] Amazon Web Services, “Amazon DynamoDB”. Accedido el 17 de septiembre de 2020. [En línea]. Disponible: https://aws.amazon.com/es/dynamodb/

[6] MDN Web Docs, “XMLHttpRequest”. Accedido el 29 de agosto de 2020. [En línea]. Disponible: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

[7] Amazon Alexa, “Host a Custom Skill as an AWS Lambda Function”. Accedido el 29 de septiembre de 2020. [En línea]. Disponible: https://developer.amazon.com/en-US/docs/alexa/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html

[8] Amazon Alexa, “Tutorial: Build an Engaging Alexa Skill”. Accedido el 7 de octubre de 2020. [En línea]. Disponible: https://developer.amazon.com/en-US/alexa/alexa-skills-kit/get-deeper/tutorials-code-samples/build-an-engaging-alexa-skill

[9] Pug, “Getting Started”. Accedido el 28 de agosto de 2020. [En línea]. Disponible: https://pugjs.org/api/getting-started.html

[10] jQuery, “jQuery.ajax”. Accedido el 28 de agosto de 2020. [En línea]. Disponible: https://api.jquery.com/jquery.ajax/