UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

Facultad de ciencias de la ingeniería

ESTUDIANTES:

Anderson Jaime

John Vera

William Vera

CARRERA:

Ingeniería de software

CURSO:

7to “A”

ASIGNATURA:

Aplicaciones distribuidas

DOCENTE:

Ing. Gleiston Guerrero

TEMA:

Las arquitecturas distribuidas

Repositorio:

https://github.com/JohnVeraXD/APP-Tareas-Capas-Azure.git

Índice

Ilustraciones

1. Introducción

En este informe, se abordará una práctica específica de Desarrollo de Aplicación en Capas, con un enfoque principal en la plataforma Firebase de Google, se examinará la teoría relacionada con otras herramientas importantes en el ámbito del desarrollo de aplicaciones en la nube, como Microsoft Azure y Amazon Web Services (AWS). El desarrollo de aplicaciones en capas es una metodología esencial que implica la división de una aplicación en componentes independientes, cada uno con una función definida. Este enfoque facilita la organización, el mantenimiento y la escalabilidad de las aplicaciones, lo que mejora su desarrollo y evolución a lo largo del tiempo, se explorarán tres temas principales [1].

En primer lugar, se examinará Firebase como una plataforma de desarrollo de aplicaciones en tiempo real, destacando su utilidad en la autenticación de usuarios, almacenamiento y sincronización de datos, así como su capacidad para implementar lógica de servidor sin servidor a través de Firebase Cloud Functions, se revisarán las herramientas ofrecidas por Microsoft Azure, una plataforma de servicios en la nube que proporciona una variedad de opciones para el desarrollo de aplicaciones modernas, desde Azure App Service para la implementación de aplicaciones web y móviles, hasta Azure Functions para la creación de funciones sin servidor y Azure SQL Database para el almacenamiento de datos. Finalmente, se analizará Amazon Web Services (AWS), otro actor importante en el campo de los servicios en la nube, con servicios como AWS Lambda para la ejecución de código sin servidor, AWS DynamoDB para el almacenamiento de datos NoSQL y AWS Cognito para la autenticación y autorización de usuarios [2].

La práctica se centrará en Firebase, comprender estas herramientas adicionales proporcionará una visión más completa del panorama actual del desarrollo de aplicaciones en la nube y las opciones disponibles para los desarrolladores en la implementación de sus proyectos.

2. Arquitectura por capas

El concepto de capas proporciona una forma conveniente de agrupar diferentes clases de arquitectura. Por ejemplo, una aplicación que se ejecuta en una sola computadora tiene una arquitectura de una sola capa (one-tier), mientras que una aplicación que se ejecuta en dos computadoras, como una aplicación web CGI que se ejecuta en un navegador (cliente) y en un servidor web, se considera una aplicación de dos capas, ya que consta de un programa cliente y un programa servidor. La diferencia principal entre ellos es que el servidor responde a las solicitudes de varios clientes, mientras que un cliente inicia solicitudes al servidor. Una aplicación de tres capas agrega un tercer programa, generalmente una base de datos, en la cual el servidor almacena información. Este tipo de aplicaciones representa una mejora incremental respecto a la arquitectura de dos capas, ya que permite que el servidor solicite o almacene información en la base de datos y luego la envíe de vuelta al cliente [3].

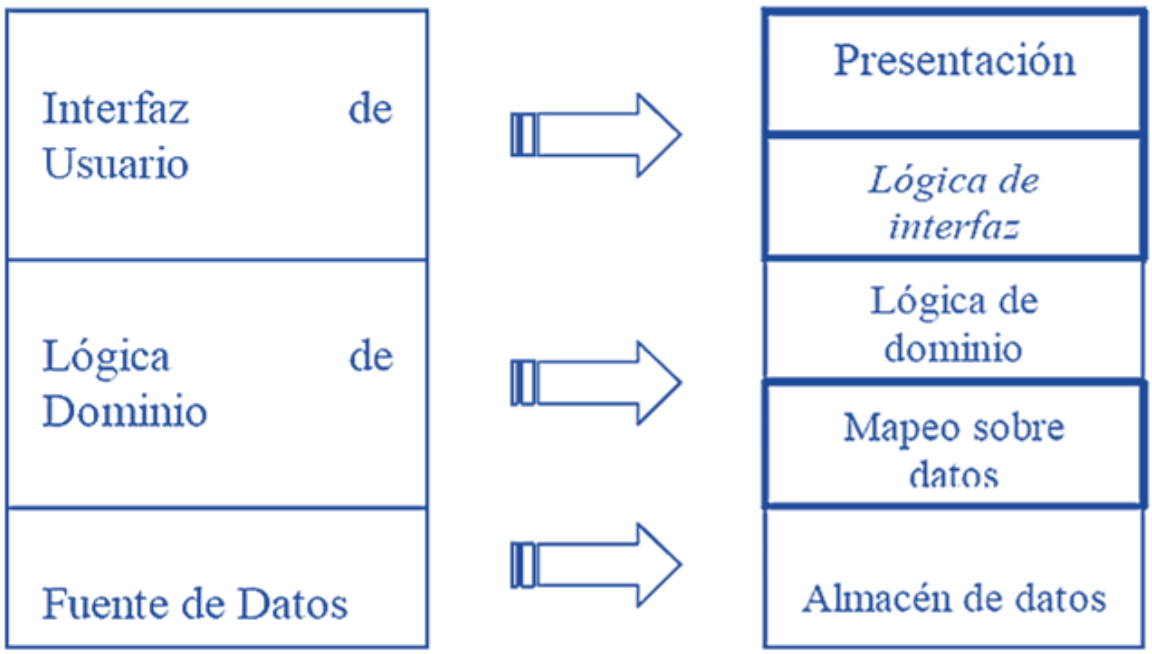


Ilustración 1 Extraído de: Evaluation of Cross-Layer Network Vulnerability of Power Communication Network Based on Multi-Dimensional and Multi-Layer Node Importance Analysis [3]

Una arquitectura de múltiples capas (n-tiers) permite que un número ilimitado de programas se ejecuten simultáneamente, enviando información de uno a otro y utilizando diferentes protocolos para comunicarse e interactuar simultáneamente. Esto facilita la creación de aplicaciones más potentes que pueden ofrecer diversos servicios a varios

clientes. Sin embargo, también presenta desventajas, como la posibilidad de introducir código maligno, como gusanos, lo que puede ocasionar problemas en el diseño, la implementación y el rendimiento. Se han desarrollado nuevas tecnologías para abordar estas preocupaciones, como CORBA, EJB, DCOM y RMI, pero saltar de una arquitectura de tres capas a una de n capas requiere considerar una serie de aspectos adicionales [4].

2.1. Elementos de la Elementos de la arquitectura en capa

Las arquitecturas en capas típicamente constan de tres componentes principales: la Capa de Presentación, la Capa de Lógica de Negocio y la Capa de Acceso a Datos. Cada una de estas capas desempeña un papel fundamental en el funcionamiento global de la aplicación y se encarga de manejar diferentes aspectos de la misma. En la capa de presentación se constituye el punto de interacción directa con el usuario. Aquí se aloja la interfaz de usuario, cuya función primordial es mostrar información y recibir las entradas del usuario. En esencia, es la parte visible y con la que el usuario interactúa directamente [5].

Dentro de la capa de Lógica de Negocio se representa el núcleo de la aplicación. Es aquí donde se lleva a cabo el procesamiento de los datos, se aplican las reglas de negocio y se toman las decisiones pertinentes. Su tarea principal consiste en definir cómo deben ser manipulados y transformados los datos antes de ser enviados a la capa de acceso a datos [6].

En la capa de acceso a datos encargada de la interacción con la base de datos o el almacenamiento subyacente, esta capa es responsable de recuperar y almacenar datos según lo requerido por la Capa de Lógica de Negocio. Su propósito principal es gestionar la comunicación con el sistema de almacenamiento de datos [7].

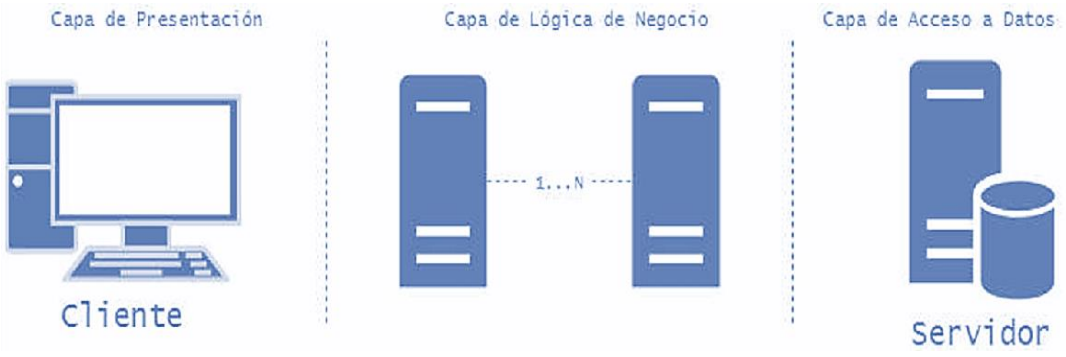


Ilustración 2 Extraído de: Transformation Architecture for Multi-Layered WebApp Source Code Generation [7]

2.2. Ventajas de la arquitectura por capas

La arquitectura por capas ofrece una serie de ventajas en el desarrollo de software. Una de las principales es la capacidad de separar claramente las responsabilidades entre las diferentes partes del sistema. Esto facilita la gestión del código, ya que cada capa se enfoca en una tarea específica, lo que permite una mejor organización y comprensión del mismo. Además, esta separación facilita la identificación y corrección de errores, ya que los problemas pueden localizarse más fácilmente en la capa correspondiente. Otra ventaja importante es la reutilización de código. Al dividir la aplicación en componentes independientes, es posible compartir y reutilizar partes del código en diferentes partes del sistema. Esto reduce la duplicación de esfuerzos y promueve la coherencia del código, ya que las funcionalidades comunes pueden ser implementadas una sola vez y utilizadas en múltiples lugares. Además, la modularidad de la arquitectura por capas facilita la actualización y evolución del sistema, ya que los cambios en una capa no necesariamente afectan a las demás. Esto permite una mayor flexibilidad y agilidad en el desarrollo de software, ya que es posible introducir nuevas características o realizar modificaciones sin afectar al funcionamiento global del sistema [8].

2.3. Desventajas de la arquitectura por capas

La arquitectura por capas también presenta ciertas desventajas que deben tenerse en cuenta. Una de ellas es la complejidad inicial de implementación. Dividir la aplicación en capas requiere una planificación cuidadosa y un diseño detallado para asegurar una distribución adecuada de las responsabilidades entre las diferentes partes del sistema. Esto puede aumentar la complejidad del desarrollo y requerir un mayor esfuerzo en la fase de diseño. Otro problema potencial es la posible sobrecarga de comunicación entre las capas. En sistemas grandes o complejos, la comunicación entre las diferentes capas puede generar una sobrecarga de comunicación y afectar el rendimiento del sistema. Esto puede ser especialmente problemático si no se optimiza adecuadamente la comunicación entre capas, lo que puede resultar en un consumo excesivo de recursos y una menor eficiencia del sistema en general [9].

3. Firebase

Firebase es la innovadora solución de Google diseñada para habilitar el funcionamiento en la nube de nuestras aplicaciones. Esta plataforma ofrece una respuesta integral a los desafíos que implica el trabajo en línea, desde el almacenamiento hasta la gestión del backend, simplificando así complejas tareas que suelen consumir un tiempo considerable en su implementación. Al basarse en la infraestructura de Google, Firebase garantiza resultados altamente confiables, seguros y escalables, lo que se traduce en un entorno propicio para el desarrollo de aplicaciones robustas y de alto rendimiento. Entre los servicios que integra, destacan la identificación de usuario, las bases de datos NoSQL y de tiempo real, la mensajería en la nube, las notificaciones push, el almacenamiento de archivos y hosting, el análisis en tiempo real del comportamiento de los usuarios, herramientas de monetización, entre otros [10].

La amplia gama de servicios abarca diversas necesidades de desarrollo y permite a los desarrolladores centrarse en la creación de experiencias de usuario excepcionales sin preocuparse por la complejidad de la infraestructura subyacente. Firebase está disponible para una variedad de sistemas, incluidos Android, iOS, Web, C++ y Unity, lo que lo convierte en una opción versátil y accesible para una amplia gama de aplicaciones. Además, el modelo de precios de Firebase ofrece una opción gratuita para un uso moderado, lo que significa que solo se paga cuando la aplicación comienza a tener éxito, lo que lo hace especialmente atractivo para desarrolladores y startups que buscan minimizar los costos iniciales [11].

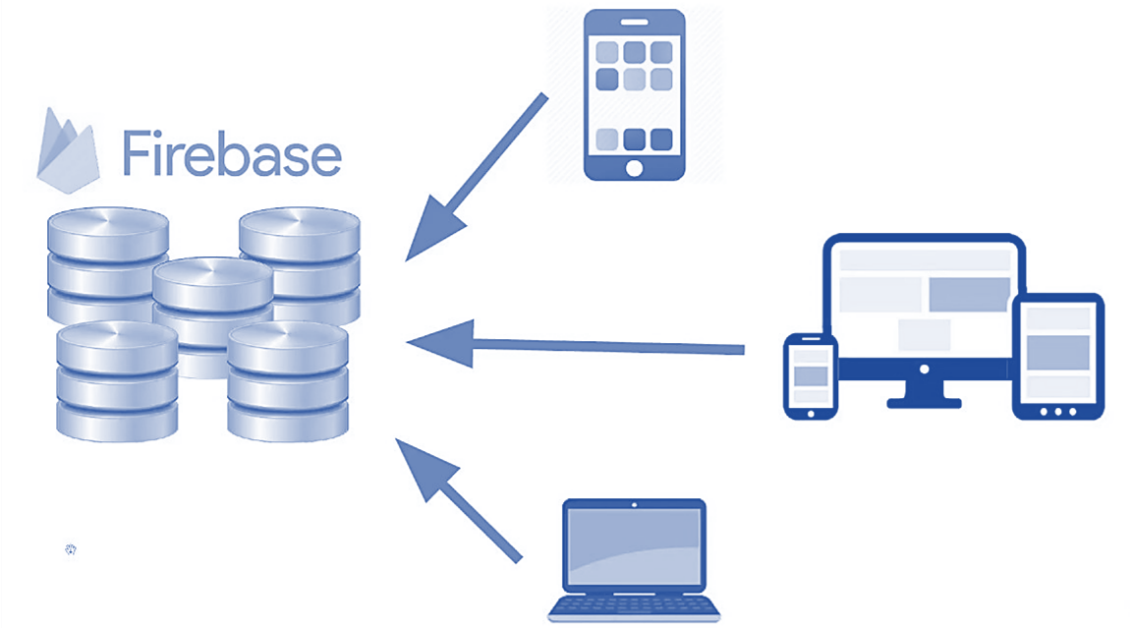


Ilustración 3 Extrido de: Firebase Database Usage and Application Technology in Modern Mobile Applications [11]

3.1. Firebase plataforma de desarrollo de aplicaciones en tiempo real

Firebase es una plataforma de desarrollo de aplicaciones en tiempo real desarrollada por Google. Esta plataforma proporciona a los desarrolladores una amplia gama de herramientas y servicios para crear aplicaciones web y móviles en tiempo real, lo que significa que los cambios realizados en la aplicación se reflejan instantáneamente en todos los dispositivos conectados. Uno de los aspectos más destacados de Firebase es su capacidad para simplificar la creación de aplicaciones en tiempo real sin necesidad de preocuparse por la configuración y gestión de la infraestructura subyacente [12].

Con Firebase, los desarrolladores pueden centrarse en la creación de características y experiencias de usuario únicas, mientras que la plataforma se encarga de aspectos como la sincronización de datos en tiempo real, la gestión de usuarios, la autenticación, el almacenamiento de datos, la mensajería y las notificaciones push, entre otros, ofrece una integración perfecta con otras herramientas y servicios de Google, lo que permite a los desarrolladores aprovechar al máximo el ecosistema de Google para crear aplicaciones potentes y escalables, proporciona a los desarrolladores las herramientas y la infraestructura necesarias para crear aplicaciones en tiempo real de manera rápida, fácil y efectiva [13].

3.2. Uso de Firebase Authentication para autenticación de usuarios

Implementa formas de autenticación de usuarios como el inicio de sesión con correo electrónico y contraseña, autenticación a través de proveedores de identidad de terceros como Google, Facebook o mediante números de teléfono, lo. Primero se debe integrar Firebase en el proyecto de Android. Luego, habilitar el método de acceso con correo electrónico y contraseña desde la sección de Autenticación en Firebase Console. se agrega la dependencia de la biblioteca de Android para Firebase Authentication en el archivo de Gradle [14].

```
dependencies {
    // Importar el BoM para la plataforma Firebase
    implementation(platform("com.google.firebase:firebase-bom:32.3.1"))
    // Agrega la dependencia para la biblioteca de Firebase Authentication
    implementation("com.google.firebase:firebase-auth-ktx")
}
```

Ilustración 4 Dependencia de firebase

Después de configurar las dependencias, se puede crear una cuenta de usuario nuevo con una contraseña. En el método onCreate de la actividad de registro, se obtiene la instancia compartida del objeto FirebaseAuth y se verifica si el usuario ya ha iniciado sesión, se utiliza el método createUserWithEmailAndPassword y pasa la dirección de correo electrónico y la contraseña del usuario nuevo [15].

```
private lateinit var mAuth: FirebaseAuth
// Inicializa Firebase Auth
mAuth = FirebaseAuth.getInstance()
// Verifica si el usuario ya ha iniciado sesión
override fun onStart() {
    super.onStart()
    val currentUser = mAuth.currentUser
    if(currentUser != null){  reload()
    }
}
// Se crea una cuenta nueva
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Si la creación de la cuenta es exitosa, actualiza la interfaz de usuario
                Log.d(TAG, "createUserWithEmail:success")
                val user = mAuth.currentUser
                updateUI(user)
            } else {
                // Si la creación de la cuenta falla, muestra un mensaje de error
                Log.w(TAG, "createUserWithEmail:failure", task.exception)
                Toast.makeText(this, "Authentication failed.", Toast.LENGTH_SHORT).show()
                updateUI(null)
            }
        }
```

Ilustración 5 Creación de usuario

Para permitir que un usuario acceda con una dirección de correo electrónico y una contraseña, los pasos son similares a los de crear una cuenta nueva. En el método onCreate de la actividad de acceso, nuevamente obtiene la instancia compartida del objeto FirebaseAuth y verifica si el usuario ya ha iniciado sesión. Luego, se utiliza el método signInWithEmailAndPassword para iniciar sesión con la dirección de correo electrónico y la contraseña proporcionadas por el usuario [16].

```
// Inicializa Firebase Auth
mAuth = FirebaseAuth.getInstance()
// Verifica si el usuario ya ha iniciado sesión
override fun onStart() {
    super.onStart()
    val currentUser = mAuth.currentUser
    if(currentUser != null){reload()
    }
}
// Inicia sesión con una dirección de correo electrónico y una contraseña
mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Si el inicio de sesión es exitoso, actualiza la interfaz de usuario
                Log.d(TAG, "signInWithEmail:success")
                val user = mAuth.currentUser
                updateUI(user)
            } else {
                // Si el inicio de sesión falla, muestra un mensaje de error
                Log.w(TAG, "signInWithEmail:failure", task.exception)
                Toast.makeText(this, "Authentication failed.", Toast.LENGTH_SHORT).show()
                updateUI(null)
            }
        }
```

Ilustración 6 Inicialización de firebase

Se recomienda habilitar la protección de enumeración de correo electrónico para mitigar riesgos potenciales relacionados con el manejo de direcciones de correo electrónico de usuarios. Una vez implementado, ya se puede gestionar la autenticación de usuarios de manera segura y eficiente en tu aplicación Android con Firebase, este procedimiento está orientado a las aplicaciones Android, de esa manera se puede realizar la autenticación [17].

3.3. Realtime Database para almacenamiento y sincronización de datos

Firebase Realtime Database es una base de datos en tiempo real alojada en la nube, diseñada para almacenar y sincronizar datos en tiempo real, permitiendo a los desarrolladores centrarse en la creación de aplicaciones sin preocuparse por la infraestructura del servidor. Entre sus características destacadas se encuentra su capacidad de sincronización en tiempo real entre todos los clientes conectados, facilitada por la tecnología de WebSockets, lo que garantiza una experiencia de usuario fluida y actualizada, utiliza una estructura de datos en forma de árbol para almacenar los datos como pares clave-valor, lo que permite una organización eficiente y la realización de consultas complejas [18].

Ofrece un sólido sistema de autenticación y control de acceso integrado, permitiendo a los desarrolladores definir reglas de seguridad personalizadas para proteger la privacidad de los datos almacenados, se integra perfectamente con otras herramientas y servicios de Firebase, como Authentication, Cloud Messaging, Storage y Hosting, lo que permite crear aplicaciones completas y escalables aprovechando las ventajas de Firebase en términos de escalabilidad, seguridad y rendimiento. Esta integración proporciona una solución integral para el desarrollo de aplicaciones modernas y eficientes [19].

```
// Configuración de Firebase
var firebaseConfig = {
apiKey: "API_KEY",
authDomain: "PROJECT_ID.firebaseapp.com",
databaseURL: "
projectId: "PROJECT_ID",
storageBucket: "PROJECT_ID.appspot.com",
messagingSenderId: "SENDER_ID",
appId: "APP_ID"
};
// Inicialización de Firebase
firebase.initializeApp(firebaseConfig);
// Obtener referencia a la base de datos
var database = firebase.database();
```

Ilustración 7 Ejemplo de cómo utilizar Firebase Realtime Database en una aplicación web

3.4. Firebase Cloud Functions para la lógica de servidor sin servidor

Diseñado para ejecutar automáticamente código de backend en respuesta a eventos activados por funciones de Firebase y solicitudes HTTPS. Este servicio te permite escribir código en JavaScript o TypeScript que se almacena y ejecuta en la nube de Google, sin la necesidad de gestionar servidores propios, se integra con la plataforma de Firebase y Google Cloud, pueden responder a una variedad de eventos generados por diferentes funciones como activadores de Firebase Authentication o Cloud Storage, permintiendo una amplia gama de casos de uso, desde la integración con servicios de terceros hasta la ejecución de lógica personalizada en respuesta a eventos específicos [20].

Ofrece un proceso de implementación sin complicaciones, se escribe y despliega el código desde la línea de comandos, de un escalado automático según los patrones de uso de los usuarios, garantiza la privacidad y seguridad de la lógica de servidor, al estar completamente aislado del cliente, puedes estar seguro de que tu código se ejecutará de manera segura y confiable sin riesgo de ingeniería inversa, después de escribir y desplegar una función, Google comienza a administrarla de inmediato. La función puede ser activada mediante solicitudes HTTP o automáticamente por eventos en la nube. A medida que la carga aumenta o disminuye, Google escala automáticamente la cantidad de instancias de servidor virtual necesarias para manejar el trabajo, asegurando un rendimiento óptimo en todo momento [21].

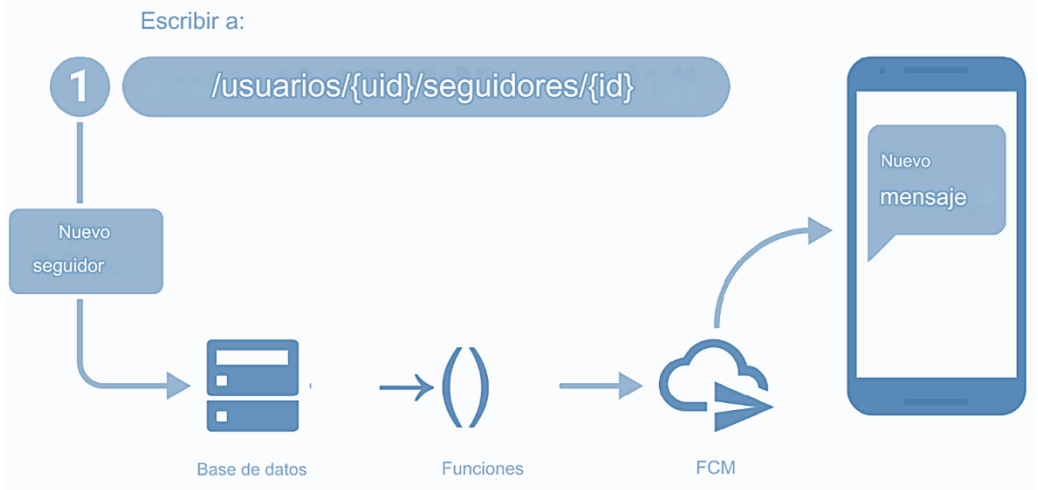


Ilustración 8 Extraído de: Using Firebase Cloud Messaging to Control Mobile Applications [21]

4. Herramientas de Desarrollo de Microsoft Azure

Ofrecen una serie de características útiles para desarrolladores, incluyendo la capacidad de realizar pruebas en producción, el escalado flexible, el uso de ranuras de implementación para desplegar y probar nuevas versiones de aplicaciones, y la conexión a recursos on-premises para una integración sin problemas. El servicio permite el escalado automático basado en la demanda y ofrece herramientas para implementación continua, lo que simplifica el proceso de desarrollo y despliegue de aplicaciones, posee la capacidad de asignar dominios personalizados y configurar certificados SSL para garantizar la seguridad y la identidad de la aplicación [22].

Dispone opciones como el uso de redes virtuales para aislar servicios de soporte y la posibilidad de utilizar Azure App Service Environment para un mayor control y escalabilidad, cuenta con herramientas de depuración, como Snapshot Debugger para .NET, que facilitan la identificación y resolución de problemas en entornos de producción, al aprovechar la plataforma totalmente administrada de Azure App Service, los desarrolladores pueden beneficiarse del parcheado automático del sistema operativo y del .NET Framework, lo que reduce la carga operativa y garantiza un entorno seguro y actualizado, en situaciones en las que se suele de necesitar un mayor control sobre la infraestructura, se dispone de opciones como Azure Virtual Machines, que permiten alojar aplicaciones en máquinas virtuales con más flexibilidad y control [23].

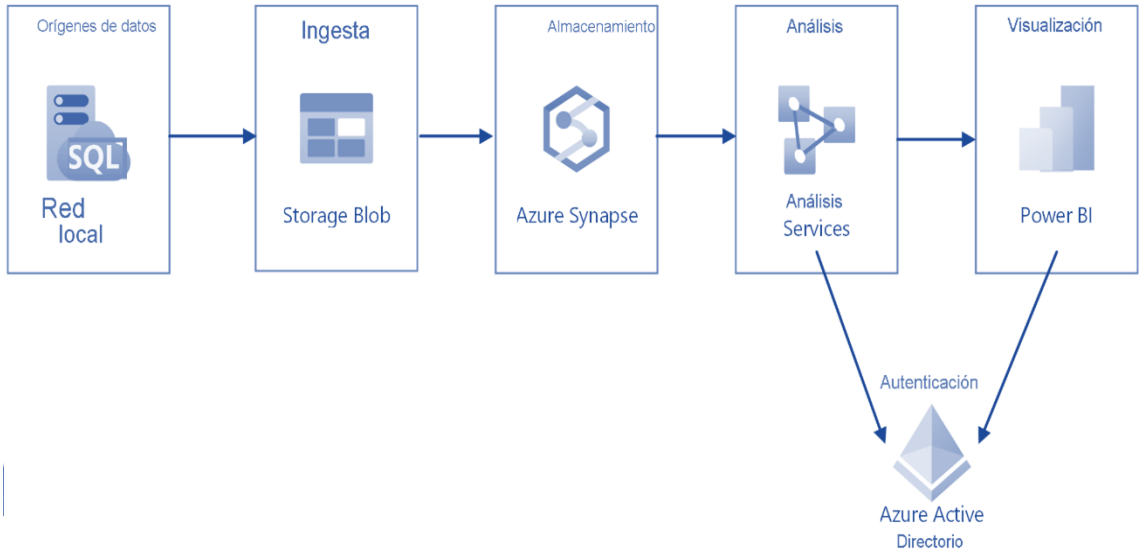


Ilustración 9 Extrañido de: Automating Tiny ML Intelligent Sensors DevOPS Using Microsoft Azure [23]

4.1. Azure como plataforma de servicios en la nube

En las aplicaciones por capas, especialmente en el ámbito del Internet de las Cosas (IoT), Azure ofrece una fácil implementación de soluciones para usar a través de Azure IoT Suite, estas soluciones preconfiguradas abordan escenarios comunes de IoT, como monitoreo remoto y mantenimiento predictivo, es personalizable y compatible con tecnologías de código abierto. La flexibilidad híbrida de Azure es una de sus características distintivas. Ofrece una nube híbrida verdaderamente consistente que permite a las empresas optimizar sus activos existentes mientras migran hacia la nube. Esta consistencia híbrida se extiende a áreas clave como el desarrollo de aplicaciones, la administración y seguridad, la gestión de identidades y la plataforma de datos. Azure incluso ofrece beneficios financieros, como el ahorro de hasta un 40% en costos de migración de máquinas virtuales de Windows Server mediante el beneficio de uso híbrido [24].

Es el único proveedor en la industria en soluciones de infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS). Los servicios de plataforma Azure PaaS pueden aumentar la productividad y el retorno de inversión, según estudios de impacto económico total realizados por firmas como Forrester. se destaca por su amplio conjunto de servicios, su flexibilidad híbrida, su reconocimiento como líder en la industria de la nube y su capacidad para ofrecer soluciones completas en áreas clave como análisis de datos y IoT. Esta combinación de características hace de Azure una opción atractiva para empresas de todos los tamaños que buscan aprovechar al máximo la tecnología en la nube [25].

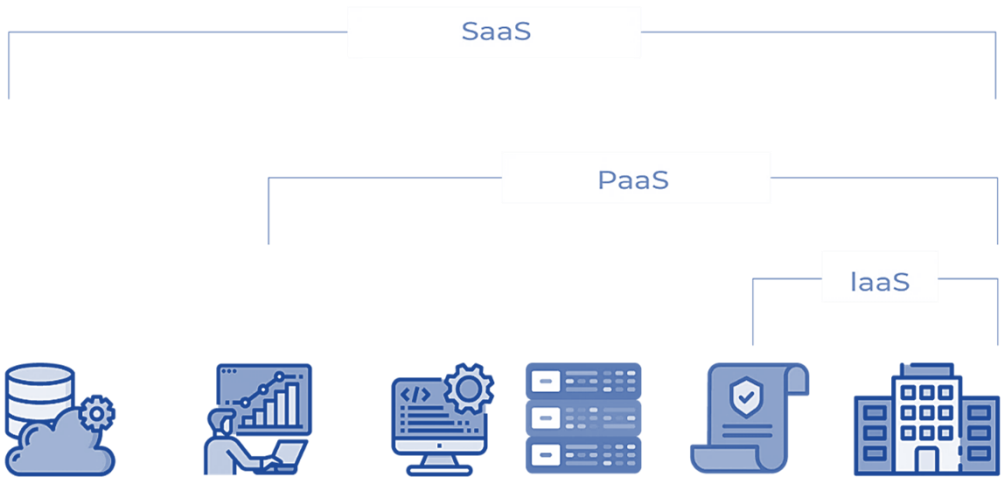
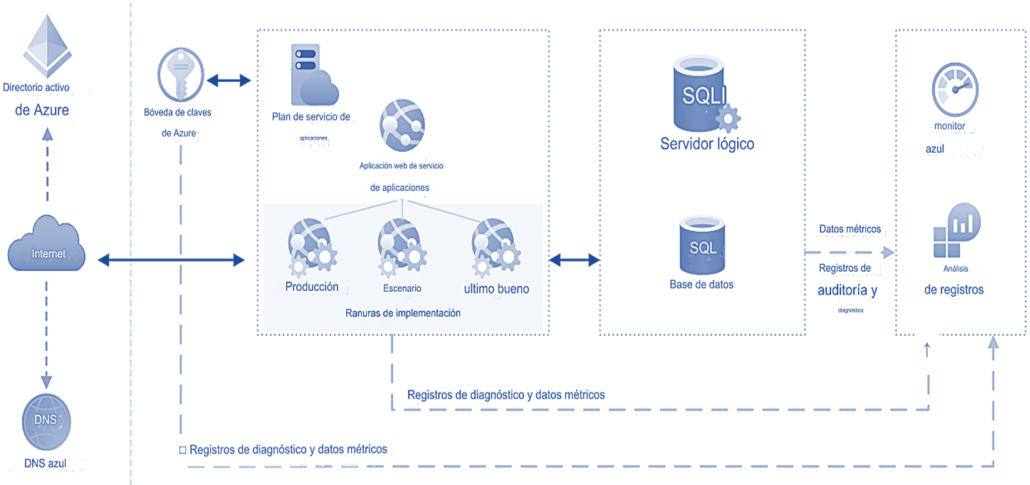


Ilustración 10 Extraído de: Creating a Mobile Application with the ESP32 Azure IoT Development Board Using a Cloud Platform [25]

4.2. Azure App Service implementación de aplicaciones web y móviles

Es una plataforma como servicio (PaaS) de Microsoft Azure, simplifica el desarrollo, implementación y escalado de aplicaciones web, móviles y API, tiene la capacidad de hospedar aplicaciones web escritas en varios lenguajes, incluyendo .NET, Java, Node.js, Python y PHP, esto se complementa con funciones como escalado automático, equilibrio de carga y alta disponibilidad, permite crear y alojar back-ends para aplicaciones móviles, ofreciendo soporte para iOS, Android y Windows, junto con características como notificaciones automáticas y autenticación. Es utilizado para la creación y alojamiento de API, con opciones para control de versiones, documentación y un portal para desarrolladores [26].

La herramienta de diseño visual de Azure Logic Apps facilita la automatización de flujos de trabajo e integración de servicios y sistemas, mientras que Azure Functions permite ejecutar código basado en eventos sin administrar la infraestructura subyacente, incluyendo Git, GitHub, Azure DevOps y contenedores de Docker, con integración sencilla en canalizaciones de integración e implementación continua (CI/CD), proporciona funciones integradas de supervisión y diagnóstico. Tiene la capacidad de escalar manual o automáticamente en función de reglas predefinidas, lo que garantiza un rendimiento óptimo y control de costos. También proporciona conectores a plataformas SaaS populares y funcionalidades de conectividad híbrida para conectar la aplicación a datos locales [27].



A continuación, se muestra un ejemplo creado por nuestra auditoria, es un ejemplo básico de una aplicación web ASP.NET Core que utiliza el enfoque MVC (Modelo-Vista-Controlador) para manejar las solicitudes HTTP, la clase Startup configura los servicios de la aplicación y define cómo se deben manejar las solicitudes HTTP, la clase Program contiene el método Main que inicia la aplicación. Este código puede ser implementado en Azure App Service para ejecutar la aplicación web en la nube [27].

```
using System;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
namespace MiAppWeb
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseRouting();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }

    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Ilustración 11 Ejemplo de una aplicación web ASP.NET Core de tipo MVC) para manejar solicitudes HTTP

4.3. Azure Functions para la creación de funciones sin servidor

Servicio de computación sin servidor ofrecido por Microsoft Azure, diseñado para acelerar y simplificar el desarrollo de aplicaciones. Funciona bajo el modelo FaaS (Function as a Service), permitiendo ejecutar pequeñas piezas de código en la nube sin preocuparse por la infraestructura, con soporte para lenguajes como C#, Java, JavaScript, Python, etc. Destaca con disparadores (triggers) y los enlaces (bindings), que permiten iniciar la ejecución de código y ofrece diversas plantillas para crear funciones, como HTTPTrigger, TimerTrigger, CosmosDBTrigger, QueueTrigger, etc. Integra servicios de Azure, como Azure Blob Storage, Azure Cosmos DB y Azure Event Grid, incluye herramientas de desarrollo como Visual Studio y el portal de Azure, y soporta la integración continua y el despliegue de código desde GitHub, Azure DevOps Services y otras plataformas [28].

Las ventajas que posee son la ejecución sin servidor, facilita el desarrollo y despliegue de código, soporta múltiples lenguajes de programación, y se activa fácilmente en base a eventos. Incluye seguridad proporcionada por Azure, integración continua, monitorización a través de Application Insights, y compatibilidad con NuGet y NPM para incorporar código de terceros. Sin embargo, también presenta desafíos, como el costo variable según el tiempo de ejecución y el plan de servicios elegido. El modelo de precios puede resultar complejo, especialmente con los planes dedicados y premium. Además, existen limitaciones en el escalado con el plan de consumo, y un timeout para la ejecución de funciones que debe ser considerado al elegir este tipo de servicio [29].

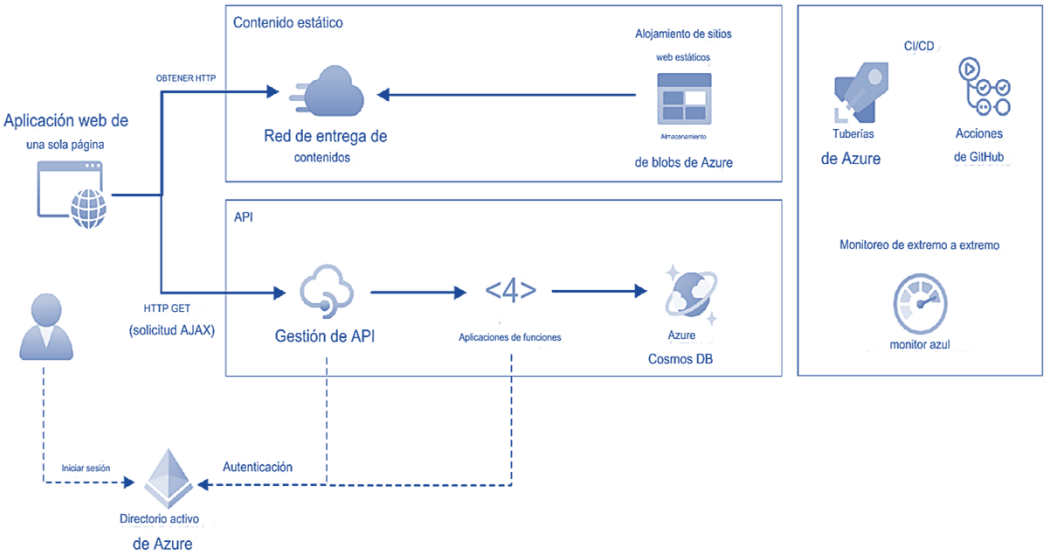


Ilustración 12 Extraído de: Deploying and Configuring Azure App Service Apps

4.4. Azure SQL Database para el almacenamiento de datos

Base de datos en la nube gestionada por parte de los servicios de Microsoft Azure, maneja funciones de gestión de bases de datos para servidores Microsoft SQL en la nube, incluyendo actualizaciones, parches, copias de seguridad y monitorización, los modelos Azure SQL Database incluyen la opción de base de datos única para necesidades aisladas y recursos dedicados, así como la opción de grupo elástico para gestionar varias bases de datos con diferentes usos y consumos, compartiendo recursos para una mayor eficiencia. Se describe como una base de datos "todo incluido" que ofrece componentes y capacidades necesarias para agilizar y acelerar el desarrollo de aplicaciones [30].

Ofrece un motor de base de datos, rendimiento altamente receptivo, escalabilidad excepcional y alta seguridad que admite construcciones de aplicaciones modernas como Geoespacial, Gráficos y JSON, expresados en el familiar lenguaje T-SQL. Destaca la importancia de la evolución de Microsoft SQL Server a lo largo de los años, con la introducción de nuevas mejoras y capacidades como Gráficos, Geoespacial, JSON, compresión de columnas, tablas optimizadas en memoria, tablas temporales, entre otros. Esta evolución ha sido fundamental a medida que Microsoft ha migrado sus servicios de cómputo y almacenamiento locales a la nube, convirtiendo a SQL Server en la base de esta transición [31].

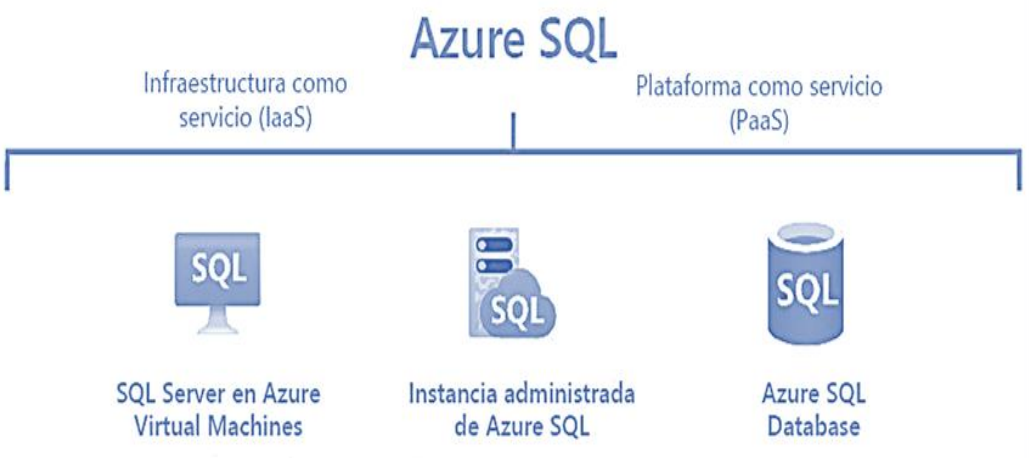


Ilustración 13 Extaído de: Automatic Database Troubleshooting of Azure SQL Databases [31]

5. Amazon Web Services (AWS)

Amazon Web Services (AWS) es una plataforma de servicios en la nube ofrecida por Amazon.com. AWS proporciona una gama de servicios de infraestructura en la nube, como cómputo, almacenamiento, bases de datos, análisis, redes, inteligencia artificial, aprendizaje automático, desarrollo de aplicaciones, seguridad, etc. Estos servicios permiten a las empresas y a los desarrolladores construir y escalar aplicaciones de manera rápida y eficiente sin la necesidad de invertir en infraestructura física. Esto permite a las empresas, start-ups y pymes, así como a los clientes del sector público, acceder a los elementos básicos que necesitan para responder de forma inmediata a los cambiantes requisitos empresariales [32].

Incluye Amazon Elastic Compute Cloud (EC2) para la computación escalable en la nube, Amazon Simple Storage Service (S3) para el almacenamiento de objetos, Amazon Relational Database Service (RDS) para bases de datos relacionales administradas, Amazon Lambda para la ejecución de funciones sin servidor, Amazon DynamoDB para bases de datos NoSQL escalables, Amazon Virtual Private Cloud (VPC) para redes privadas virtuales, y Amazon Elastic Container Service (ECS) para la gestión de contenedores Docker. Contiene una gama de herramientas de gestión y monitoreo, seguridad y cumplimiento, así como servicios de inteligencia artificial y aprendizaje automático a través de AWS AI/ML [33].

Ilustración 14 Extraído de: Amaznon Web Service [33]

5.1. AWS como plataforma de servicios en la nube

AWS es una plataforma cloud que ofrece desde tecnologías de infraestructura como cómputo, almacenamiento y bases de datos hasta tecnologías emergentes como inteligencia artificial y Blockchain. Dispone de una amplia gama de soluciones para satisfacer las necesidades de las empresas en la nube. Por ejemplo, Amazon EC2 (Amazon Elastic Compute Cloud) es un servicio que permite aumentar recursos de cómputo de manera escalable en la nube, brinda la flexibilidad necesaria para manejar picos repentinos en la carga del servidor [34].

Amazon S3 (Amazon Simple Storage Service) proporciona un almacenamiento en la nube escalable, permite almacenar una cantidad ilimitada de datos de manera segura y accesible desde cualquier lugar, Amazon CloudFront, un servicio CDN (Content Delivery Network) reduce el tiempo de carga al distribuir contenido a través de una red global de Edge Locations, Amazon RDS (Servicio de base de datos relacional) simplifica la gestión y escalabilidad de bases de datos relacionales en la nube, mientras que Amazon VPC (Virtual Private Cloud) permite crear redes privadas aisladas dentro de la plataforma, ofreciendo control sobre el entorno virtual. Amazon Lambda ejecuta código basado en eventos sin administrar la infraestructura subyacente [35].

Ilustración 15 Extraído de: Containerized Application for IoT Devices: Comparison between balenaCloud and Amazon Web Services Approaches [35]

5.2. AWS Lambda para la ejecución de código sin servidor

AWS Lambda es un servicio de computación sin servidor ofrecido por Amazon Web Services, permite ejecutar código en respuesta a eventos específicos, como cambios en datos, actualizaciones de bases de datos o solicitudes HTTP, sin la necesidad de provisionar servidores, cada vez que ocurre un evento, Lambda se encarga de la ejecución del código en la nube, son los pilares de este servicio, donde los eventos actúan como desencadenantes y las funciones Lambda representan el código que se ejecuta en respuesta a esos eventos, admite varios lenguajes de programación, incluidos Node.js, Python, Java, C#, Go y Ruby, lo que brinda flexibilidad, ofrece integración con otros servicios de AWS, lo que permite crear aplicaciones sin servidor que aprovechan, se pueden crear flujos de trabajo complejos utilizando Lambda en combinación con servicios como Amazon S3, DynamoDB, API Gateway, SNS (Amazon Simple Notification Service) y etc [36].

Al ser un servicio sin servidor, elimina la necesidad de administrar servidores, se encarga de gestionar la infraestructura, Lambda ofrece un modelo de precios basado en el uso, donde solo se paga por el tiempo de ejecución de su código, sin costos fijos ni cargos por tiempo de inactividad, permite uso eficiente de recursos y un ahorro de costos significativos, se integra perfectamente con otros servicios de AWS, lo que facilita la construcción de aplicaciones, se puede utilizar junto con servicios como Amazon S3, DynamoDB, SNS, SQS, API Gateway, etc. [37].

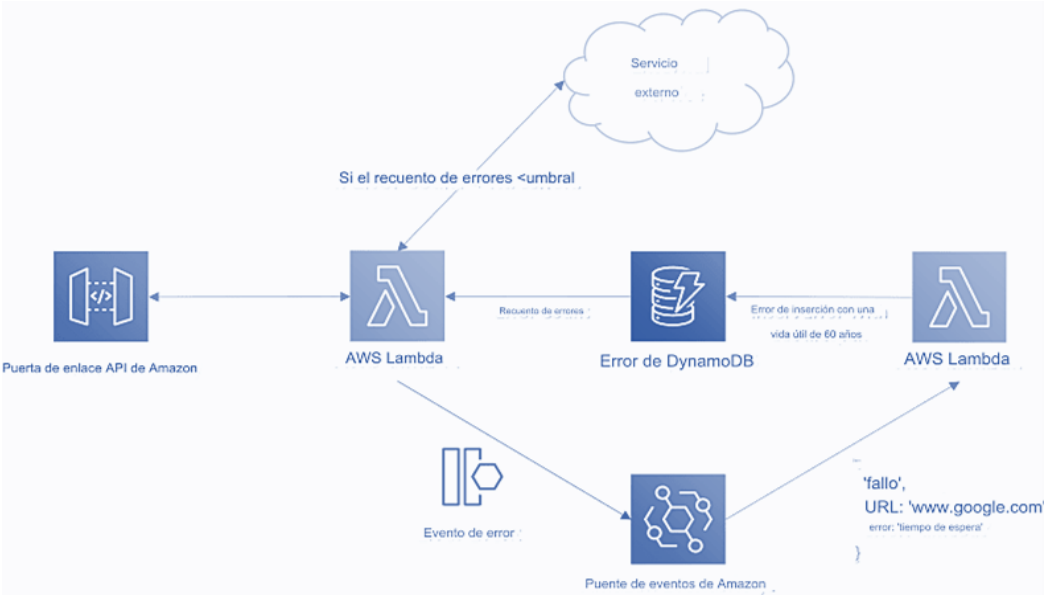


Ilustración 16 Extraído de: Effect of optimizing Java deployment artifacts on AWS Lambda

5.3. AWS DynamoDB para el almacenamiento de datos NoSQL

Es una base de datos NoSQL de clave-valor sin servidor y completamente administrada que está diseñada para ejecutar aplicaciones de alto rendimiento a cualquier escala, ofrece seguridad integrada, copias de seguridad continuas, replicación automatizada en varias regiones, almacenamiento de caché en memoria y herramientas de importación y exportación de datos, tiene la capacidad para escalar de forma automática y transparente según la demanda de tráfico se pueden ajustar la capacidad de lectura y escritura de sus tablas de manera independiente por lo de se encarga de distribuir automáticamente la carga [38]

Ofrece una baja latencia de acceso a los datos, para aplicaciones que requieren respuestas rápidas, como aplicaciones web, móviles y de juegos, utiliza índices secundarios globales y locales para admitir consultas eficientes y permite realizar operaciones de transacciones atómicas en varios elementos de datos dentro de una tabla, proporciona características de seguridad avanzadas, el cifrado de datos en reposo y en tránsito, la autenticación basada en IAM (Identity and Access Management), y la integración con AWS CloudTrail para el registro de auditoría de actividades, se puede monitorear el rendimiento de sus tablas utilizando métricas y alarmas integradas para realizar copias de seguridad y restauraciones de datos de forma flexible según sea necesario [39].

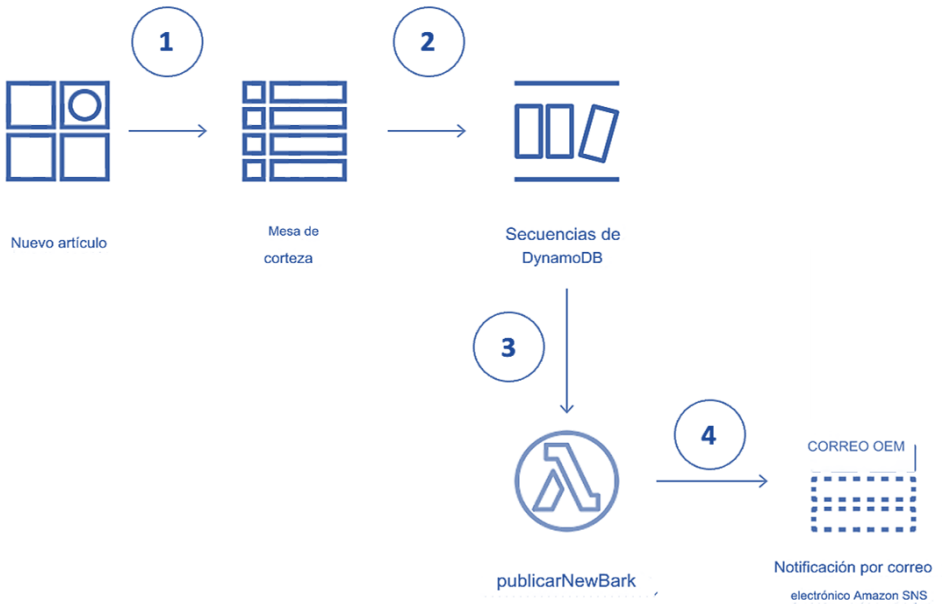


Ilustración 17Extraído de: Performance Improvement on a Learning Assessment Web Application Using AWS DynamoDB as a Cache Database [38].

5.4. AWS Cognito para la autenticación y autorización de usuarios

Es un servicio de Amazon Web Services que proporciona identidad como servicio (IDaaS) y facilita la autenticación y autorización en aplicaciones, es más allá de usuarios de IAM (Identidad y Gestión de Acceso), ofreciendo una plataforma completa para la gestión de la identidad en aplicaciones web y móviles, incluye la capacidad de gestionar el acceso de usuarios a través de varios proveedores de identidad, como redes sociales y servicios de AWS y de autenticación multifactor (MFA) para una mayor seguridad, permite organizar a los usuarios en grupos y ofrece credenciales temporales, su escalabilidad y flexibilidad maneja un aumento en el número de usuarios o aplicaciones [40].

Los casos de uso comunes incluyen aplicaciones web y móviles que requieren un inicio de sesión seguro, así como entornos IoT que necesitan manejar identidades de dispositivos para un acceso seguro, se usa en entornos multi-inquilino donde cada "inquilino" puede tener su propio grupo de usuarios y políticas, comparado con Azure Active Directory B2C de Microsoft, Amazon Cognito se diferencia por la gestión de identidad y acceso para aplicaciones, ofreciendo funciones de sincronización de datos en dispositivos y autenticación para dispositivos, incluye su integración profunda con la infraestructura de AWS y su capacidad para personalizar la experiencia de autenticación y registro de usuarios [41].

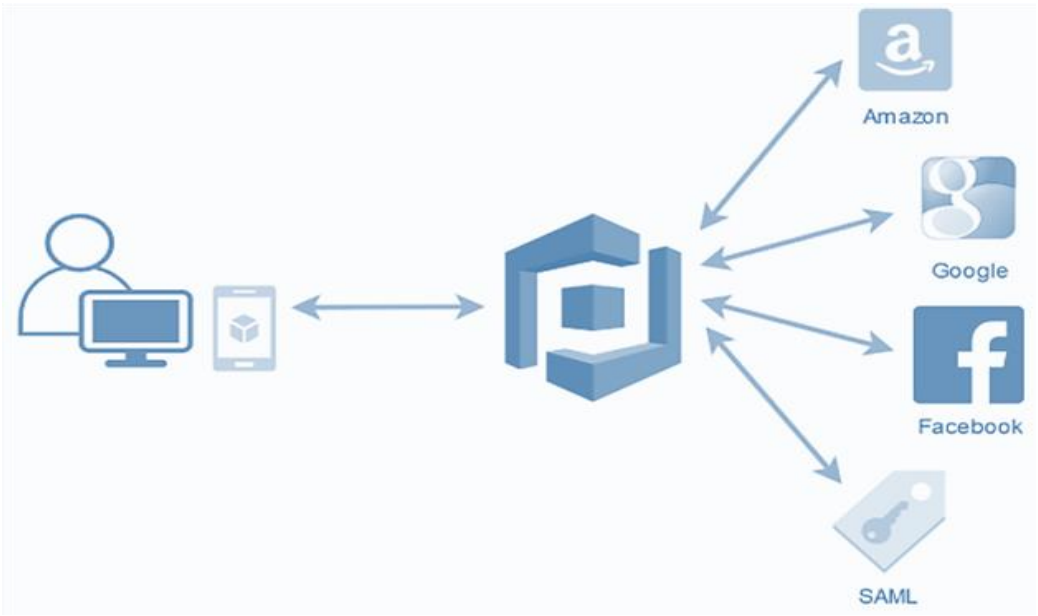


Ilustración 18 Extraído de: Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT on AWS [41]

6. Desarrollo de la práctica

Para esta práctica se realizó una aplicación web usando la arquitectura de aplicaciones en capas, para la gestión de tareas mediante grupos. También el uso de los servicio de Microsoft Azure para desplegar la base de datos en la nube.

6.1. Planteamiento del problema

Aplicación Web: Gestión Eficiente de Tareas

En la dinámica diaria de proyectos, metas o trabajos, la organización y gestión de tareas juega un papel crucial. La falta de una herramienta eficiente puede resultar en una pérdida de productividad y dificultades en la colaboración. Por lo tanto, surge la necesidad de una aplicación web que aborde de manera integral la gestión de tareas, permitiendo a los usuarios organizar sus actividades de manera efectiva y facilitar la colaboración en entornos grupales.

El propósito fundamental de la aplicación es proporcionar a los usuarios una herramienta eficiente para organizar y gestionar tareas. Además, busca fomentar la colaboración efectiva en grupos, proyectos o trabajos, ofreciendo una plataforma centralizada y fácil de usar.

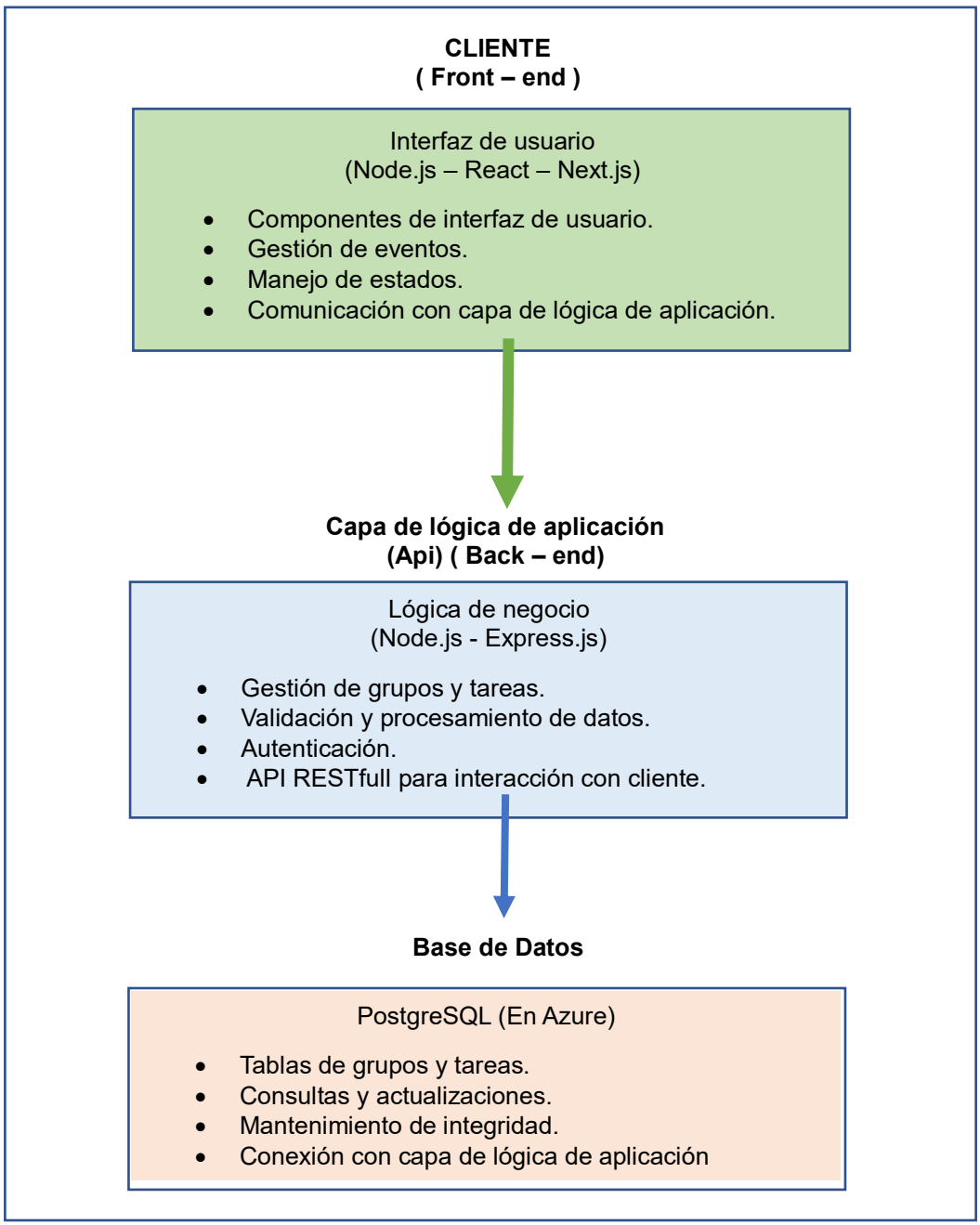
6.2. Funcionalidades importantes

1. **Creación de Grupos:** Permite a los usuarios organizar tareas relacionadas en grupos, facilitando la gestión de proyectos o actividades con múltiples componentes.
2. **Añadir Tareas:** Proporciona la capacidad de agregar tareas con detalles específicos, como título, descripción, fecha de creación y prioridad. Esto asegura una captura detallada de la información esencial para cada tarea.
3. **Modificación de Tareas:** Ofrece funciones de edición para ajustar la información de las tareas existentes, incluyendo la descripción y otros detalles relevantes. Esto permite una adaptabilidad continua a medida que evolucionan los proyectos.
4. **Marcado de Tareas como Completadas:** Permite a los usuarios indicar la finalización de tareas, proporcionando una visión clara del progreso y logros alcanzados en el desarrollo de proyectos o trabajos.

5. **Selección de Tareas Importantes:** Introduce la capacidad de destacar tareas importantes, ofreciendo una manera efectiva de priorizar y resaltar elementos cruciales en medio de otras actividades.

6.3. App usando la arquitectura de tres capas

Ahora se mostrará un diagrama representativo de la arquitectura de la aplicación desarrollada a través de una estructura de tres capas, basándonos en la conceptualización mostrada anteriormente, se han considerado Frameworks que se ha ajustan a nuestras necesidades funcionales.



6.4. Creación de la capa de presentación (cliente)

Como ya se mencionó previamente se está haciendo uso de la arquitectura en capas para la creación de la APP de gestión de tareas, a continuación, se detalla todo el proceso de desarrollo y los pasos que se siguieron para desarrollar la capa de presentación (cliente).

Paso 1: Instalación de las tecnologías necesarias para el desarrollo:

El cliente web (Front-end) se realizó con ayuda de Node.js. También, se hizo uso del framework nextjs y a su vez el uso de la biblioteca de React con ayuda del framework Tailwind CSS para el diseño de páginas web.

Siguiendo la siguiente estructura para el cliente web:

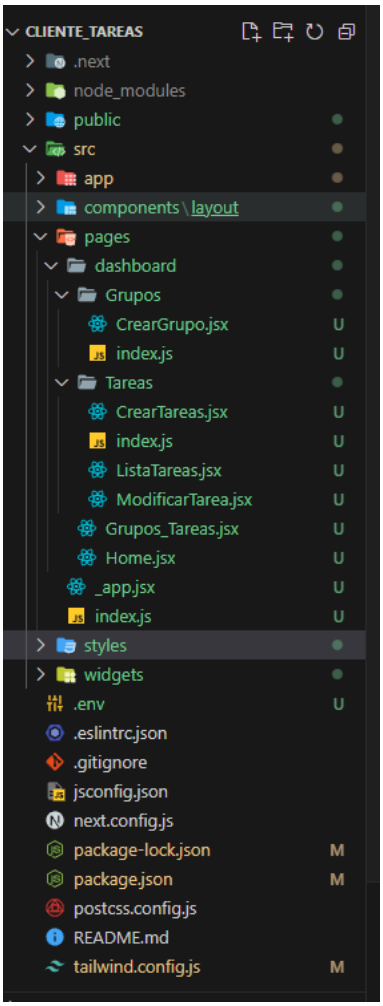


Ilustración 19. Estructura del cliente Web.

Paso 2: Desarrollo la interfaz principal (autenticación con google):

Desarrollo de la interfaz principal en donde se mostrara el inicio de sesión con google como un método de autenticación y haciendo uso del jsonwebtoken para la autenticación mediante cookies. (index.js)

```
export default function Index() {
    //Borrar cookies en caso de existir alguna
    useEffect(() => {
        const cookies = new Cookies();
        cookies.remove("id_user");
        cookies.remove("myTokenName");
    }, []);
    //variables para el inicio de sesion
    const [user, setUser] = useState({
        email: "",
        password: "",
    });
    //variable para detectar un error y mostrar el error
    const [error, setError] = useState(false);
    //variable para almacenar el mensaje del error
    const [mensajeError, setMensajeError] = useState('');
    //variable para saber si se inicio sesion correctamente
    const [autenticado, setAutencidado] = useState(false);
    //funcion para alimentar la variable que contiene las credenciales
    const HandleChange = (e) => {
        setUser({ ...user, [e.target.name]: e.target.value });
    };
    //Envento clik para iniciar con google
    const login = useGoogleLogin({
        onSuccess: async (respose) => {
            try {
                const res = await axios.get(
                    "https://www.googleapis.com/oauth2/v3/userinfo",
                    {
                        headers: {
                            Authorization: `Bearer ${respose.access_token}`,
                        },
                    }
                );

                console.log(res.data);
                //Aqui va para sacar el token ty sacar el mail del token que te regresa google
                const email = res.data.email;
                console.log(email);

                //Llama al metodo pasandole el email
                GoogleLogin(email);

            } catch (error) {
                console.log(error);

            }
        },
    });
    const GoogleLogin = async (p_email) => {
        try {

            console.log(p_email);
            const result = await axios.post(
                process.env.NEXT_PUBLIC_ACCESLINK + "authgoogle/LoginGoogle",
                { p_email },
                {
                    withCredentials: true,
                }
            );
            //console.log("asdas", result);
            const cookies = new Cookies();
            //Cookie para el token
            cookies.set("myTokenName", result.data.token, { path: "/" }); //enviar cokiee y almacenarla
            //Cookie para el id del usuario
            cookies.set("id_user", result.data.id, { path: "/" });
            //para abrir la nueva ruta en la misma pestana
            Router.push("/dashboard/Home");
            console.log(result.data);
        } catch (error) {
            console.log(error);
            //colocar una alerta de error cuando no se pueda inciar sesion
            setError(true);
            setMensajeError(error.response.data.error);
        }
    };
    //funcion para cerrar el dialog del error
    const cerrar = (valor) => {
        setError(valor)
    }
    return (
        <div className=" w-full h-ful ">
            {error ? <Dialog_Error mensaje={mensajeError} titulo="Error Inicio de sesion" cerrar={cerrar} /> : ("")}
            <Card color="transparent" shadow={false} className="mx-auto w-full max-w-[24rem] mt-10 shadow-xl p-6 hover:shadow-green-500 border-4
border-blue-900  border-solid text-center bg-white items-center justify-center rounded-none">
                <div className="p-2 mx-auto">
                    <Typography variant="h4" color="blue-gray">
                        Gestionador de Tareas
                    </Typography>
                    <Lottie animationData={anim} className="w-40 mx-auto" />
                </div>
                <Typography variant="h4" color="blue-gray">
                    Iniciar Sesion
```

Ilustración 20. Código para la Página principal autenticación con google.

A continuación se muestra como se visualiza la interfaz para el usuario.

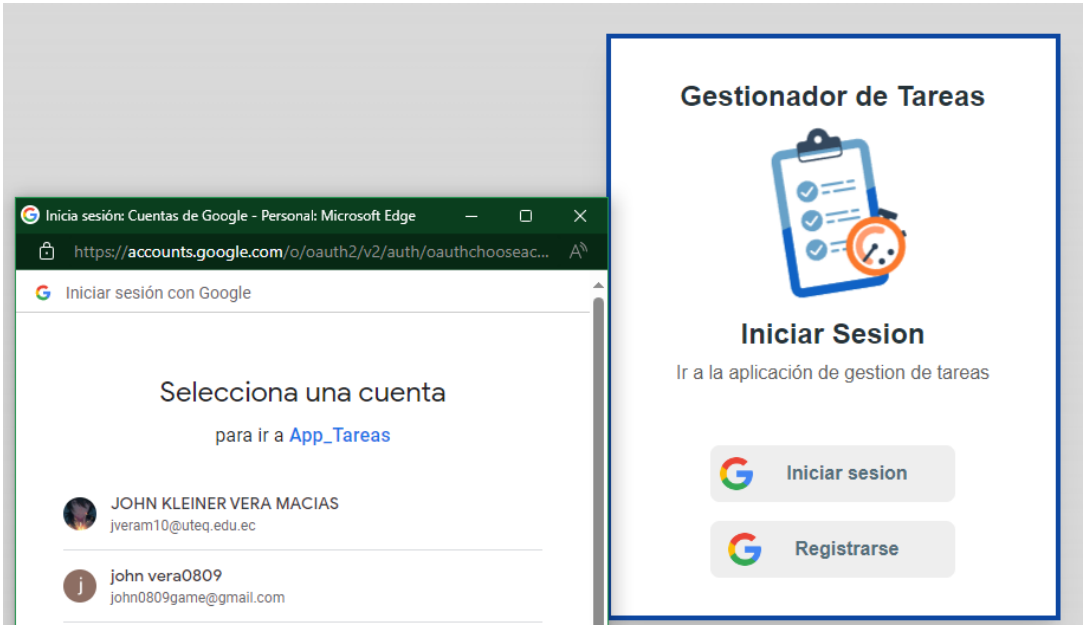


Ilustración 21. Interfaz principal (Autenticación con google).

Paso 2: Desarrollo la página principal (Home.jsx):

Desarrollo de la interfaz principal en donde se mostrara la página principal con la barra de navegación para acceder a las diferentes funciones de la app (Home.jsx)

```jsx
import React from 'react'
import BarraNavegacion from '@/components/layout/BarraNavegacion'
import Footer from '@/components/layout/Footer'
import '../../styles/globals.css'
import SimpleNavbar from '@/components/layout/SimpleNavar'

export default function Home() {
  return (
    <div className="contenedor">
      <SimpleNavbar />

      <div className="bg-white contenido border border-gray-300 p-2 mt-2 ml-2 mr-2">
        <h1>Hola como estas</h1>
      </div>

      <div className="p-2">
        <Footer />
      </div>
    </div>
  )
}
```

Ilustración 22. Código de la página principal.

A continuación se muestra como se visualiza la interfaz para el usuario.

Ilustración 23. Interfaz principal.

Paso 3: Desarrollo la página principal para la tareas (ListarTareas.jsx):

Desarrollo de la interfaz principal en donde se mostrara la página principal con la barra de navegación para acceder a las diferentes funciones de la app (Home.jsx)

```
return (
        <div className="p-1 contenedor">
            <SimpleNavbar />

            <div className=" bg-white contenido border border-gray-300 p-2 mt-2 ml-2 mr-2">

                <Card className="h-full w-full mt-1 rounded-none p-0">
                    {<CrearTareas re_id_grupo={r_id_grupo} abrir={openCreate} cerrar={cerrar} crear={crear} />}
                    {<ModificarTarea re_titulo={r1_titulo} re_descripcion={r1_descripcion} re_importante={r1_importante} re_id_tareas={r1_id_tareas}
abrirM={openCreateM} cerrarM={cerrarM} crearM={crearM} />}
                    <CardHeader floated={false} shadow={false} className="rounded-none">
                        <div className="mb-8 flex items-center justify-between gap-8">
                            <div>
                                <Typography variant="h5" color="blue-gray">
                                    Lista de Tareas creadas.
                                </Typography>
                            </div>
                            <div className="flex shrink-0 flex-col gap-2 sm:flex-row">
                                <Button
                                    className="flex items-center gap-3"
                                    size="sm"
                                    color="green"
                                    onClick={() => setOpenCreate(true)}
                                >
                                    <PlusIcon strokeWidth={2} className="h-4 w-4" /> Crear Tareas
                                </Button>
                            </div>
                        </div>
                    </CardHeader>
                    <CardBody className="overflow-scroll px-0">
                        {secciones.length === 0 && (
                            <Typography
                                color="gray"
                                variant="h4"
                                className="mt-1 font-normal mx-auto items-center text-center"
                            >
                                Usted no tiene ninguna tarea creada
                            </Typography>
                        )}
                        <div className="grid grid-cols-1 md:grid-cols-3 gap-3 p-5">
                            {secciones.map(
                                ({ r_id_tareas, r_titulo, r_descripcion, r_fecha_creacion, r_completada, r_importante }) => (
                                    <div
                                        key={r_id_tareas}
                                        className={`${r_importante
                                            ? ' bg-blue-200'  // Se aplica si r_importante es true
                                            : 'bg-blue-gray-50'
                                            } shadow-2xl rounded-none cursor-pointer hover:border-4 `}
                                        onClick={() => { ModificarTareas(r_titulo, r_descripcion, r_importante, r_id_tareas); setOpenCreateM(true) }}
                                    >
                                        <div className="bg-zinc-900 rounded-2xl">
                                            <div className="mx-auto">
                                                <div className="w-full p-4">
                                                    <input
                                                        className={`${r_importante
                                                            ? ' bg-blue-200'  // Se aplica si r_importante es true
                                                            : 'bg-blue-gray-50'
                                                            }  w-full text-2xl text-center font-semibol text-blue-gray-800 `}
                                                        disabled
                                                        value={r_titulo}
                                                    />
                                                </div>
                                                <div className="w-full p-4">
                                                    <input
                                                        className={`${r_importante
```

6.5. Creación de la capa lógica de negocio (Api)

Como ya se mencionó previamente se está haciendo uso de la arquitectura en capas para la creación de la APP de gestión de tareas, a continuación, se detalla todo el proceso de desarrollo y los pasos que se siguieron para desarrollar la capa de lógica de negocio (API).

Paso 1: Instalación de las tecnologías necesarias para el desarrollo:

La API se realizó con ayuda de Node.js, también se utilizó el framework Express, este proporciona mecanismos para: Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).

Siguiendo la siguiente estructura para la Api:

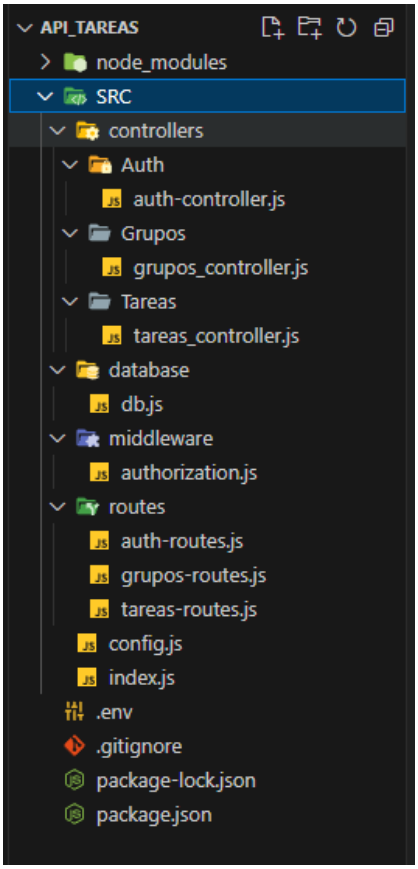


Ilustración 24. Estructura de la Api.

Paso 2: Desarrollo del servidor (index.js):

Desarrollo de la implementación del servidor en el archivo index.js donde está toda la lógica para iniciar el servidor, también se realiza las configuraciones necesarias para el cord, las rutas principales públicas y las que necesitan autorización mediante el JSON Web Token.

```
const express = require("express");
const cors = require("cors");
const dotenv = require("dotenv");
const cookie_parser = require("cookie-parser");
const http = require("http");

//Este middleware se ejecuta antes de entrar a una ruta protegida, es decir, se necesita un
token valido para acceder
const { authenticateToken } = require("./middleware/authorization.js");

//Importacion de las rutas principales en variables
const authRoutes = require("./routes/auth-routes.js");
const gruposRoutes = require("./routes/grupos-routes.js");
const tareasRoutes = require("./routes/tareas-routes.js");

//config entorno
dotenv.config();
```

```
//Configurar el puerto donde se abre la API
const app = express();
const PORT = 4099;

//direccion donde se abre
const corsOptions = {credentials: true, origin: "http://localhost:3000"};

//configuracion del server
app.use(cors(corsOptions));
app.use(express.json());
app.use(cookie_parser());

//Rutas publicas
app.use("/authgoogle", authRoutes);
```

```
//Rutas protegidas por el token
//app.use("/users", authenticateToken, userRoutes);
app.use("/grupos", authenticateToken, gruposRoutes);
app.use("/tareas", authenticateToken, tareasRoutes);
```

```
//Iniciar la API
app.listen(PORT, () => console.log("SERVER ON PORT" + PORT));
```

Ilustración 25. Código para la implantación del servidor.

Paso 3: Creación de rutas para la autenticación con google (auth-routes.js):

Desarrollo de las rutas para el acceso a la autenticación con google:

```
const { Router } = require('express');
const router = Router();

const {iniciarUserGoogle } = require('../controllers/Auth/auth-controller');

router.post('/LoginGoogle',iniciarUserGoogle);

module.exports = router;
```

Ilustración 26. Código para las rutas de inicio de sesión.

Paso 4: Creación del controlador para la autenticación con google (auth-controller.js):

Desarrollo del controlador para la gestión de los métodos encargados de realizar la lógica, gestión y la conexión a la base de datos respectiva para autenticación Google, y la asignación del token si esta es válida:

```
const pool = require('../../database/db');
const jwt = require('jsonwebtoken');
const { serialize } = require('cookie');

//VerficarUsuario con el correo que devuelve google y otorgar token

const iniciarUserGoogle = async (req, res, next) => {
    try {
        const { p_email } = req.body;
        const users = await pool.query('select * from verification_google($1)', [p_email]);
        let verification = users.rows[0];
        //Extraer el resultado del bool para saber si el login es correcto
        let result = verification.verification;
        //Si el Login fallo es decir es diferente del estado 1
        if (result != 1) return res.status(401).json({ error: verification.mensaje });
        //Si no entonces se le otorga un token xd
        const token = jwt.sign({
            exp: Math.floor(Date.now() / 1000) + 60 * 60 * 24 * 30,
            p_email: p_email,
        }, 'SECRET') //el secret deberia estan en el .env

        const serialized = serialize('myTokenName', token, {
            httpOnly: true,
            secure: process.env.NODE_ENV === 'production',
            sameSite: 'none',
            maxAge: 1000 * 60 * 2, //dos minutos para hacer las pruebas
            path: '/'
        })
        //maxAge: 1000 * 60 * 60 * 24 * 30, //30 dias
        //1000 * 60 * 15,  // 15 minutos
        res.setHeader('Set-Cookie', serialized)
        //Guardar el id del usuario en el json
        //Ver si el usuario es admin general y guardar en json para que se guarde como cookie
        const data_auth = await pool.query('select  * from auth_data($1)', [p_email]);
        //parsear los data_auth para enviar en un solo json
        let data = data_auth.rows[0];
        let userc = data.userc;
        //let isadmin = data.verification;
        //Ver si el usuario es admin de area y guardar en json para que se guarde como cookie
        return res.json({ verification: "true", token: token, id: userc });
    } catch (error) {
        next(error);
    }
}
module.exports = {
    iniciarUserGoogle
};
```

Ilustración 27. Código del controlador para la autenticación con google.

Paso 5: Creación del middleware para la autenticación del token (authorization.js):

Desarrollo del middleware en donde se crea el archivo authorization.js, donde se desarrolla la lógica necesaria para la comprobación del token, validando si este es válido y otorgar los permisos.

```
const jwt = require('jsonwebtoken');

function authenticateToken(req, res, next) {
    const { myTokenName } = req.cookies;
    if (myTokenName == null) return res.status(401).json({ error: "Null Token" });
    jwt.verify(myTokenName, 'SECRET', (error, user) => {
        if (error) return res.status(403).json({ error: error.message });
        req.user = user;
        next();
    })
}

//EL SECRET DEBE ESTAR EN UN .ENV

module.exports = {
    authenticateToken
};
```

Ilustración 28. Código del middleware.

Paso 6: Creación de rutas para las tareas (tareas -routes.js):

Desarrollo de las rutas mediante los métodos (get, post, put y delete) para la gestión de las tareas en la aplicación:

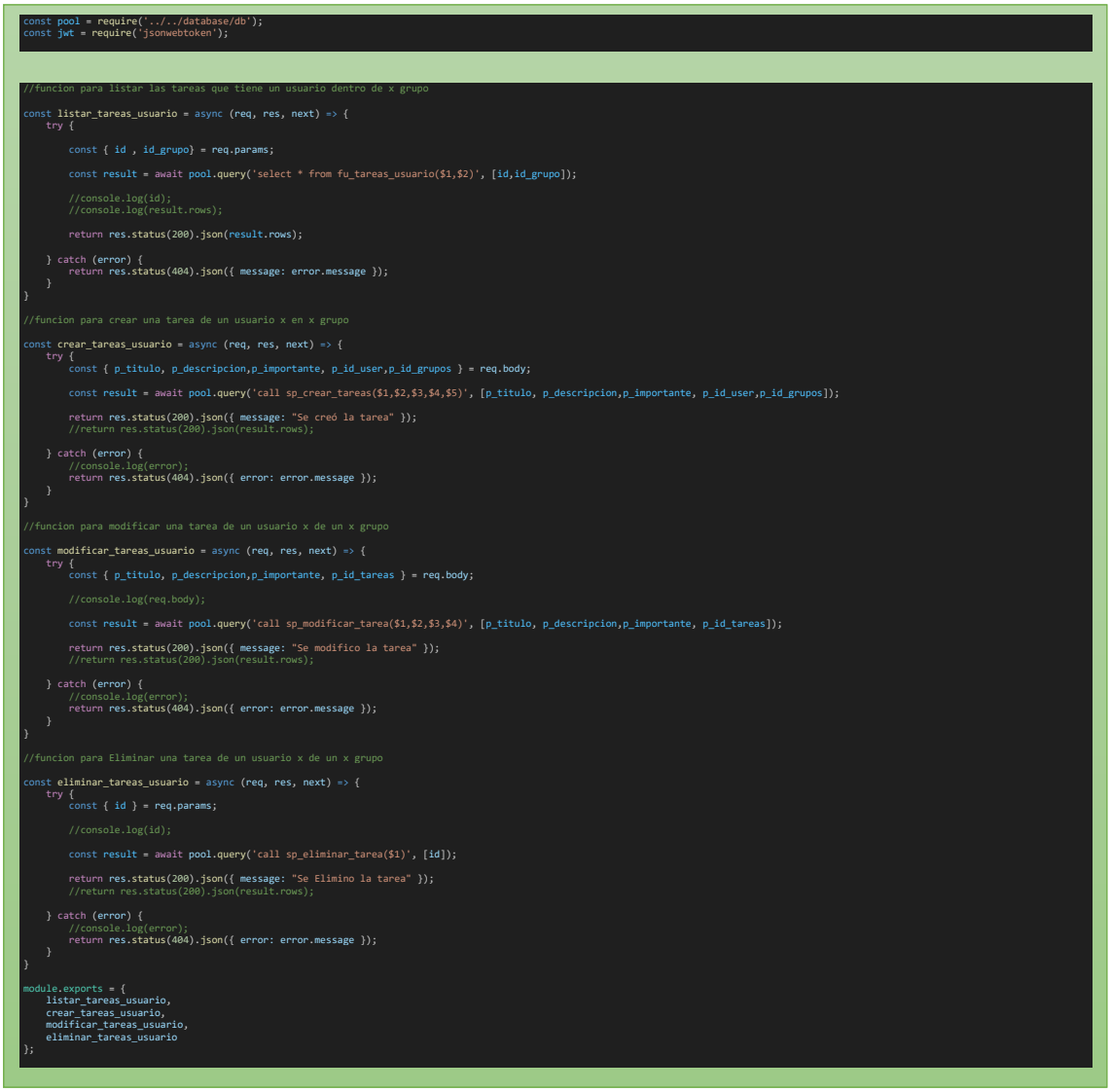
```
const { Router } = require('express');
const router = Router();
const {listar_tareas_usuario,crear_tareas_usuario ,
modificar_tareas_usuario,eliminar_tareas_usuario} =
require('../controllers/Tareas/tareas_controller');

//Obtener recursos
router.get('/ListaTareas/:id/:id_grupo', listar_tareas_usuario);
//Crear recursos
router.post('/CrearTarea', crear_tareas_usuario);
//Editar recursos
router.put('/ModificarTarea',modificar_tareas_usuario);
//Eliminar recursos
router.delete('/EliminarTarea/:id',eliminar_tareas_usuario);
module.exports = router;
```

Ilustración 29. Código de las rutas para las tareas.

Paso 7: Creación del controlador (tareas_controller.js):

Desarrollo del controlador para la gestión de los métodos encargados de realizar la lógica, manipulación de los datos, gestión y la conexión a la base de datos respectiva para la gestión de las tareas en la aplicación:



```
const pool = require('../../database/db');
const jwt = require('jsonwebtoken');


//funcion para listar las tareas que tiene un usuario dentro de x grupo

const listar_tareas_usuario = async (req, res, next) => {
    try {

        const { id , id_grupo} = req.params;

        const result = await pool.query('select * from fu_tareas_usuario($1,$2)', [id,id_grupo]);

        //console.log(id);
        //console.log(result.rows);

        return res.status(200).json(result.rows);

    } catch (error) {
        return res.status(404).json({ message: error.message });
    }
}

//funcion para crear una tarea de un usuario x en x grupo

const crear_tareas_usuario = async (req, res, next) => {
    try {
        const { p_titulo, p_descripcion,p_importante, p_id_user,p_id_grupos } = req.body;

        const result = await pool.query('call sp_crear_tareas($1,$2,$3,$4,$5)', [p_titulo, p_descripcion,p_importante, p_id_user,p_id_grupos]);

        return res.status(200).json({ message: "Se creó la tarea" });
        //return res.status(200).json(result.rows);

    } catch (error) {
        //console.log(error);
        return res.status(404).json({ error: error.message });
    }
}

//funcion para modificar una tarea de un usuario x de un x grupo

const modificar_tareas_usuario = async (req, res, next) => {
    try {
        const { p_titulo, p_descripcion,p_importante, p_id_tareas } = req.body;

        //console.log(req.body);

        const result = await pool.query('call sp_modificar_tarea($1,$2,$3,$4)', [p_titulo, p_descripcion,p_importante, p_id_tareas]);

        return res.status(200).json({ message: "Se modifico la tarea" });
        //return res.status(200).json(result.rows);

    } catch (error) {
        //console.log(error);
        return res.status(404).json({ error: error.message });
    }
}

//funcion para Eliminar una tarea de un usuario x de un x grupo

const eliminar_tareas_usuario = async (req, res, next) => {
    try {
        const { id } = req.params;

        //console.log(id);

        const result = await pool.query('call sp_eliminar_tarea($1)', [id]);

        return res.status(200).json({ message: "Se Elimino la tarea" });
        //return res.status(200).json(result.rows);

    } catch (error) {
        //console.log(error);
        return res.status(404).json({ error: error.message });
    }
}

module.exports = {
    listar_tareas_usuario,
    crear_tareas_usuario,
    modificar_tareas_usuario,
    eliminar_tareas_usuario
};
```

Ilustración 30. Código del controlador para las tareas.

6.6. Creación de la capa de persistencia (Base Azure PostgreSQL)

Como ya se mencionó previamente se está haciendo uso de la arquitectura en capas para la creación de la APP de gestión de tareas, a continuación, se detalla todo el proceso de desarrollo y los pasos que se siguieron para desarrollar la capa de persitencia (PostgreSQL con Azure).

Paso 1: Registro en Microsoft Azure para la creación de la BD:

Para la creación de la base de datos, utilizamos el servicio que nos ofrece Microsoft Azure para desplegar una base de datos de PostgreSQL en la nube.

Primero nos dirigimos a la página oficial de Microsoft Azure, nos registramos y completamos los pasos que nos piden, una vez registrados nos redirige a la página principal:

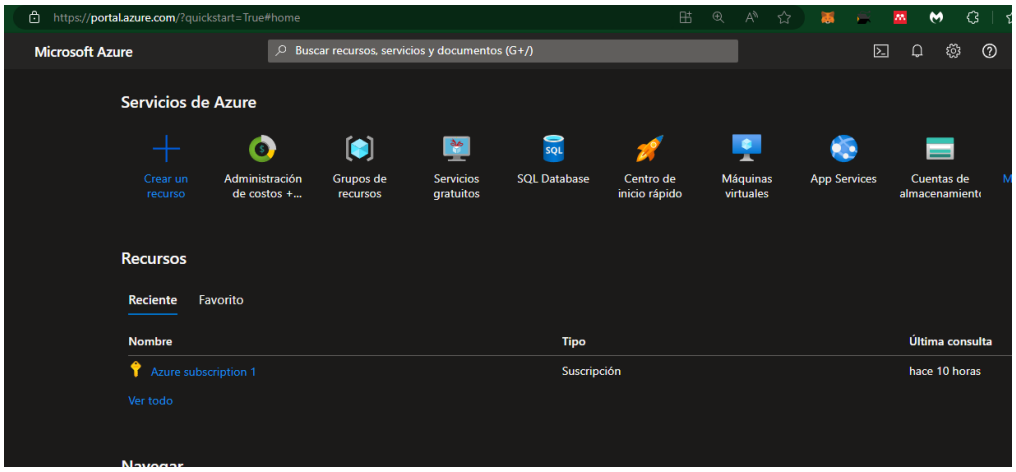


Ilustración 31. Página principal de Microsoft Azure.

Paso 2: Uso de los servicios gratuitos de Azure:

Nos dirigimos a los servicios gratuitos que nos ofrece Azure:

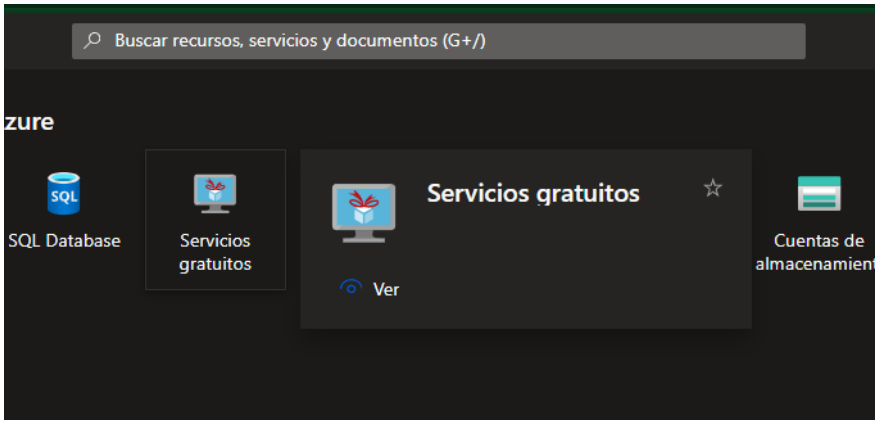


Ilustración 32. Servicios gratuitos de Azure.

Paso 3: Creación de la base de datos con PostgreSQL en Azure:

Después de dirigirnos a los servicios gratuitos, nos abre una página en donde se muestran todos los servicios que podemos usar, en este caso usamos el servicio de base de datos flexible para PostgreSQL:

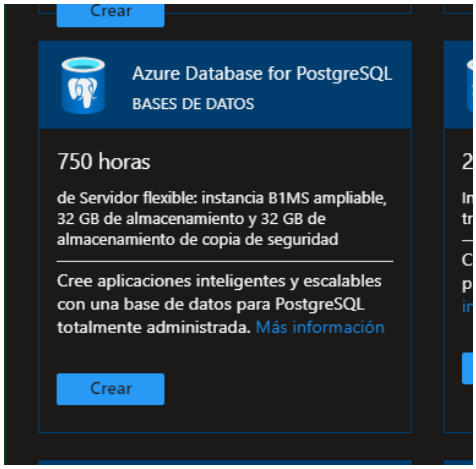


Ilustración 33. Base de datos PostgreSQL en Azure.

Paso 4: Datos para la base de datos PostgreSQL:

Después nos abre la página en donde nos pide los datos necesarios para la creación de la base de datos, donde rellenamos los campos correspondientes:

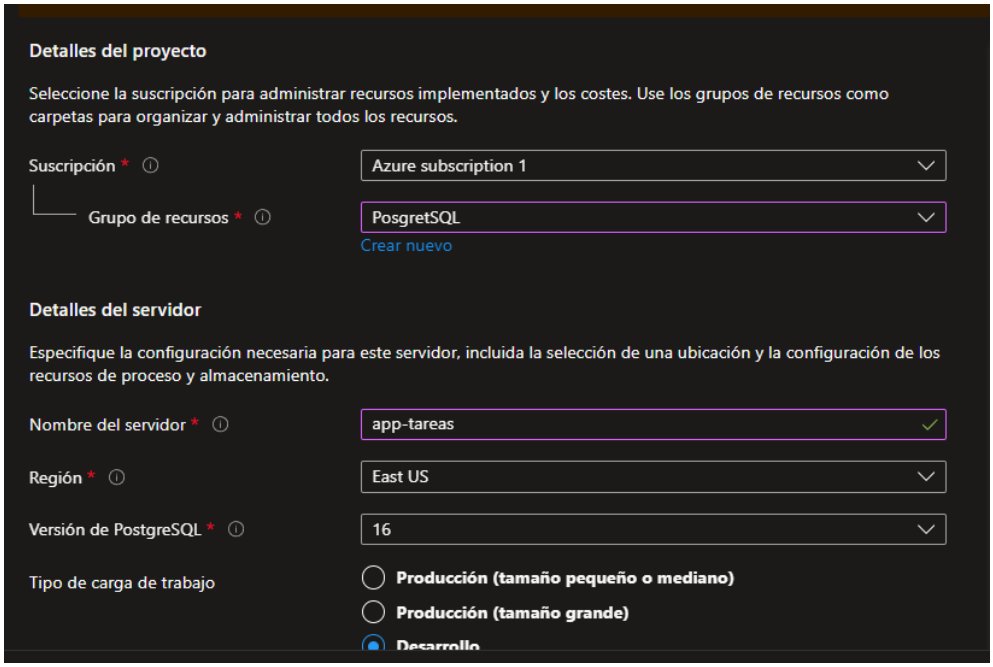


Ilustración 34. Formulario para los datos de la BD.

También nos pide el usuario y la contraseña respectivamente para poder acceder a la base de datos:

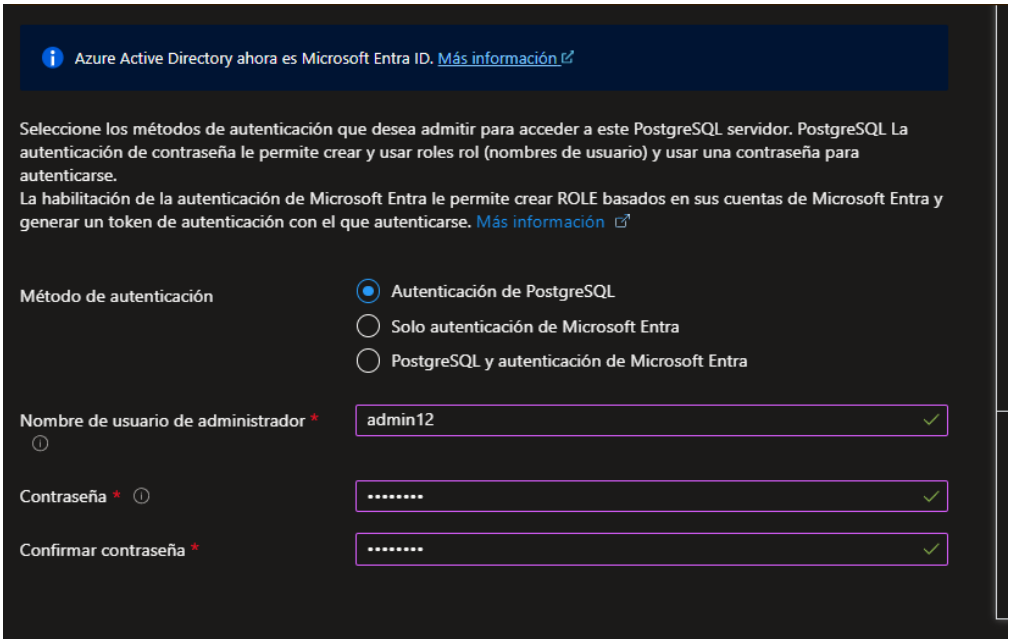


Ilustración 35. Formulario para los datos de la BD.

Por ultimo debemos configurar las reglas del firewall, en donde agregamos la dirección IP con la que vamos a realizar la conexión respectiva, esto nos lo pide ya que como estamos usando la base de datos en modo de desarrollo tenemos que otorgar los permisos necesarios.

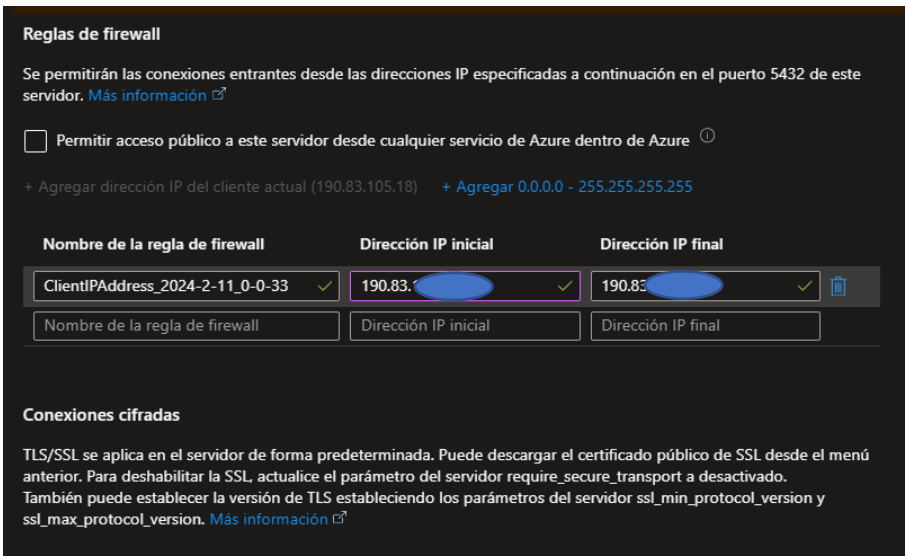


Ilustración 36. Configuración del firewall.

Paso 5: Obtención de los datos de la BD creada:

Después de la creación de la BD, nos redirige a la página en donde se muestra la información necesaria para poder establecer la conexión con la BD.

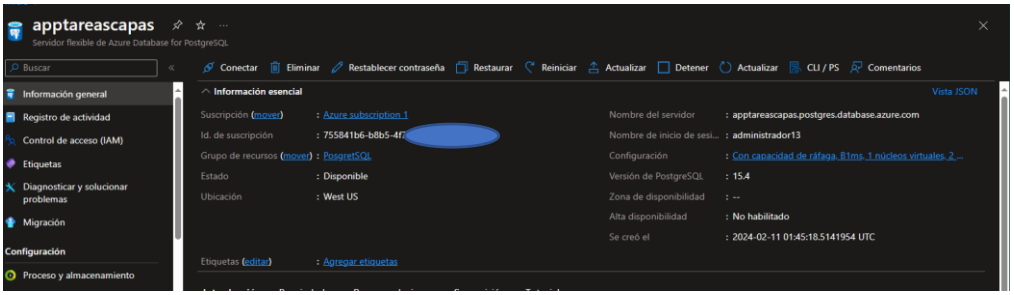


Ilustración 37. Datos de la BD.

Aquí nos da el servidor (host): apptareascapas.postgres.database.azure.com

También el usuario: administrador13

Paso 6: Conexión a la base de datos con PgAdmin:

Con la base de datos creada en la nube necesitamos acceder a ella para poder manipularla, en donde vamos a crear las tablas, procedimientos, funciones necesarias para la manipulación de los datos.

Nos dirigimos al PgAdmin, y creamos un nuevo grupo de server, creamos un grupo llamado Azure. Dentro de este registramos un nuevo servidor le ponemos PostgreSQL.

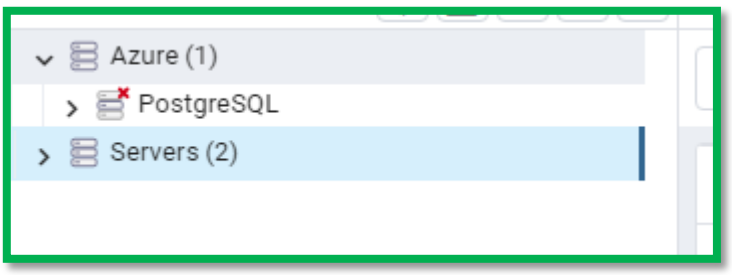


Ilustración 38. Conexión con PgAdmin.

En las opciones para la creación del servidor ponemos los datos que nos proporcionó Azure para poder estableces la conexión.

Host: apptareascapas.postgres.database.azure.com

Port: 5432

Username: administrador13

Password: $123John

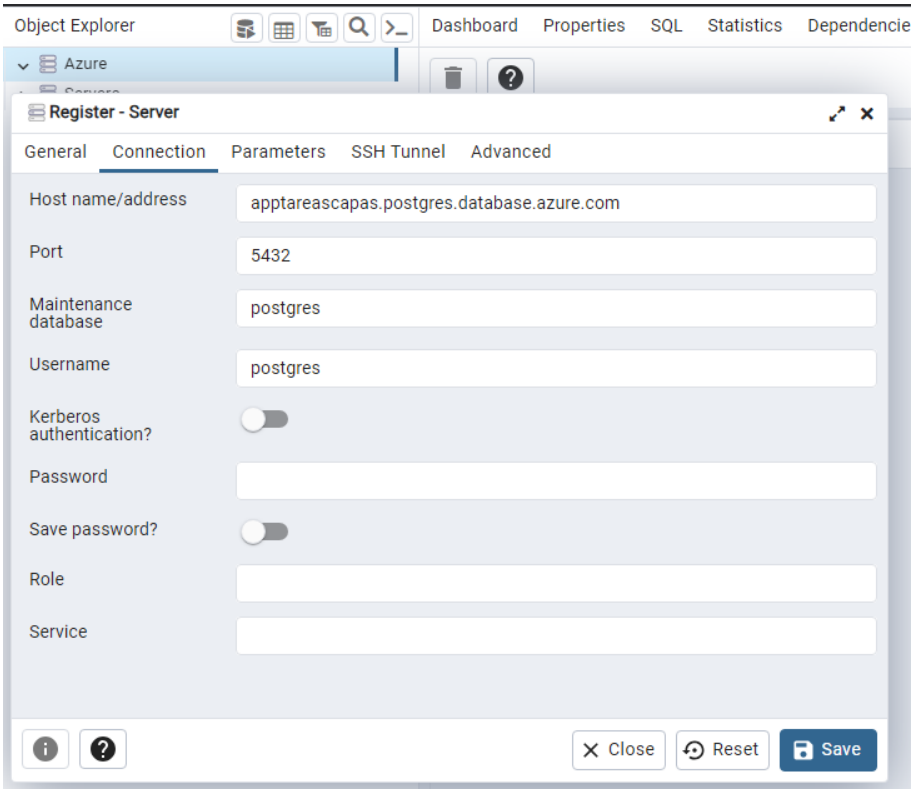


Ilustración 39. Registro de la base de datos creada en Azure.

Paso 7: Crear las tablas y procedimientos:

Una vez que tenemos la base de datos conectada, creamos una base de datos donde vamos a crear las tablas y procedimientos respectivos.

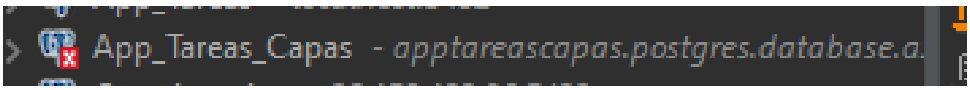


Ilustración 40. Base de datos creada.

```sql
CREATE TABLE usuarios (
    id_usuarios int generated always as identity,
    nombre_Apellidos VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    google_id VARCHAR(100) unique,
    estado bool Default true not null,
    fecha_creacion TIMESTAMP NULL DEFAULT now(),
    Primary Key(id_usuarios)
);

--drop table tareas

CREATE TABLE tareas (
    id_tareas int generated always as identity,
    titulo VARCHAR(100) not null UNIQUE,
    descripcion VARCHAR(100) not null,
    fecha_creacion TIMESTAMP NULL DEFAULT now(),
    completada BOOLEAN DEFAULT FALSE,
    importante BOOLEAN DEFAULT FALSE, -- Nueva columna para marcar si la tarea es
importante
    id_usuarios int REFERENCES usuarios(id_usuarios) ON DELETE cascade,
    id_grupos int REFERENCES grupos(id_grupos) ON DELETE cascade,
    estado bool default true,
    Primary Key(id_tareas)
);

--drop table grupos

CREATE TABLE grupos(
    id_grupos int generated always as identity,
    fecha_creacion TIMESTAMP NULL DEFAULT now(),
    titulo VARCHAR(100) not null UNIQUE,
    descripcion VARCHAR(100) not null,
    id_usuarios int REFERENCES usuarios(id_usuarios) ON DELETE cascade,
    estado bool default true,
    Primary Key(id_grupos)
);
```

Ilustración 41. Código para la creación de las tablas.

Paso 8: Conexión de la Api con la base de datos en la nube de Azure:

En el archivo .env establecemos los parámetros necesarios para poder establecer la conexión con la base de datos que se encuentra en la nube de Azure:

```
DB_USER=administrador13
DB_PASSWORD=$123John
DB_HOST=apptareascapas.postgres.database.azure.com
DB_DATABASE=App_Tareas_Capas
DB_PORT=5432
```

Ilustración 42. Configuración del archivo .env en la API.

7. Conclusión

Esta investigación ha resaltado el papel esencial de tecnologías como Microsoft Azure en el desarrollo de aplicaciones modernas y para abordar los desafíos asociados con la distribución de aplicaciones. Además, la comparación directa entre Firebase y Microsoft Azure ha resaltado las diferencias en sus enfoques y funcionalidades, lo que proporciona una base sólida para la toma de decisiones en cuanto a la selección de la plataforma más adecuada para proyectos futuros. Si bien Firebase ofrece una integración sencilla con productos de Google y una rápida puesta en marcha, Microsoft Azure destaca por su escalabilidad, seguridad avanzada y una amplia gama de servicios que pueden adaptarse a las necesidades específicas de cualquier aplicación.

Además, se ha llevado a cabo una práctica específica que implica el desarrollo de una aplicación denominada "Task Manager". Esta aplicación, creada utilizando Azure, incluyó la integración de la autenticación de Google y otras funcionalidades relacionadas con la gestión de tareas. Esta práctica ha permitido aplicar los conceptos teóricos discutidos en esta investigación, demostrando la eficacia y versatilidad de Microsoft Azure como plataforma de desarrollo de aplicaciones en tiempo real. Esta integración entre la teoría y la práctica es fundamental para comprender a fondo las complejidades y ventajas que ofrecen las arquitecturas distribuidas en el contexto actual de la informática y el desarrollo de aplicaciones.

8. Bibliografía

[1] W. Jin, J. Lee, W. Jang, S. Kim, H. Park, y J. W. Lee, «DRAM Translation Layer: Software-Transparent DRAM Power Savings for Disaggregated Memory», en *Proceedings - International Symposium on Computer Architecture*, Institute of Electrical and Electronics Engineers Inc., jun. 2023, pp. 217-229. doi: 10.1145/3579371.3589051.

[2] W. Wang, K. Lin, y J. Zhao, «Network-Based Layered Architecture for Long-Term Prediction», *IEEE Access*, vol. 8, pp. 18252-18257, 2020, doi: 10.1109/ACCESS.2020.2968473.

[3] Z. Min, W. Muqing, Q. Lilin, A. Quanbiao, y L. Sixu, «Evaluation of Cross-Layer Network Vulnerability of Power Communication Network Based on Multi-Dimensional and Multi-Layer Node Importance Analysis», *IEEE Access*, vol. 10, pp. 67181-67197, 2022, doi: 10.1109/ACCESS.2021.3109902.

[4] Z. Wang, S. K. Oh, E. H. Kim, Z. Fu, y W. Pedrycz, «Hierarchically Reorganized Multi-Layer Fuzzy Neural Networks Architecture Driven With the Aid of Node Selection Strategies and Structural Network Optimization», *IEEE Access*, vol. 10, pp. 7772-7792, 2022, doi: 10.1109/ACCESS.2022.3140397.

[5] W. Alhakami, H. S. El-Sayed, O. S. Faragallah, y M. G. El-Mashed, «Efficient Security Architecture for Physical Layer in mmWave Communication Systems», *IEEE Access*, vol. 10, pp. 113923-113934, 2022, doi: 10.1109/ACCESS.2022.3217244.

[6] N. M. Yungaicela-Naula, C. Vargas-Rosales, y J. A. Perez-Diaz, «SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning», *IEEE Access*, vol. 9, pp. 108495-108512, 2021, doi: 10.1109/ACCESS.2021.3101650.

[7] R. Tesoriero, A. Rueda, J. A. Gallud, M. D. Lozano, y A. Fernando, «Transformation Architecture for Multi-Layered WebApp Source Code Generation», *IEEE Access*, vol. 10, pp. 5223-5237, 2022, doi: 10.1109/ACCESS.2022.3141702.

[8] H. Moayed y E. G. Mansoori, «Skipout: An Adaptive Layer-Level Regularization Framework for Deep Neural Networks», *IEEE Access*, vol. 10, pp. 62391-62401, 2022, doi: 10.1109/ACCESS.2022.3178091.

[9] M. Whaiduzzaman, M. J. N. Mahi, A. Barros, M. I. Khalil, C. Fidge, y R. Buyya, «BFIM: Performance Measurement of a Blockchain Based Hierarchical Tree Layered Fog-IoT Microservice Architecture», *IEEE Access*, vol. 9, pp. 106655-106674, 2021, doi: 10.1109/ACCESS.2021.3100072.

[10] W. -J. Li, C. Yen, Y. -S. Lin, S. -C. Tung and S. Huang, "JustIoT Internet of Things based on the Firebase real-time database," 2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE), Hsinchu, Taiwan, 2019, pp. 43-47, doi: 10.1109/SMILE.2018.8353979.

[11] U. A. Madaminov and M. R. Allaberganova, "Firebase Database Usage and Application Technology in Modern Mobile Applications," 2023 IEEE XVI International Scientific and Technical Conference Actual Problems of Electronic Instrument Engineering (APEIE), Novosibirsk, Russian Federation, 2023, pp. 1690-1694, doi: 10.1109/APEIE59731.2023.10347828.

[12] K. S. Lakshmi, C. L. P. S. Sudha, Y. V. Bharathi, M. Divyanjali and M. Suneetha, "Android Application on e-notifier for College Events Management using Firebase," 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2021, pp. 1-5, doi: 10.1109/ICECA52323.2021.9676084.

[13] S. Pathima, G. SindhuPriya, G. L. Yesaswini and S. Suhasini, "Digitalization of Catalogue Automation System with Firebase," 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 2023, pp. 1-7, doi: 10.1109/INCET57972.2023.10170052.

[14] H. K. Abdulla, H. K. Omar, A. S. H. Abdul-Qawy and A. O. Ali, "IoT-Driven Smart Car Integration with Google Firebase for Empowering Premises Security," 2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 2023, pp. 387-394, doi: 10.1109/MIUCC58832.2023.10278358.

[15] A. Alsalemi et al., "Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation," 2019 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 2019, pp. 178-182, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.31.

[16] A. Kumar Sharma and L. Mohan Saini, "IoT based Diagnosing Myocardial Infarction through Firebase Web Application," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 190-195, doi: 10.1109/ICECA.2019.8822150.

[17] V. Margapuri, N. Penumajji and M. Neilsen, "PiBase: An IoT-based Security System Using Google Firebase and Raspberry Pi," 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), Bandung, Indonesia, 2021, pp. 79-85, doi: 10.1109/IoTaIS53735.2021.9628513.

[18] H. Kimm and D. Flynn, "Android Application for Tracking Pedestrian Movements in Realtime with Firebase," 2022 IEEE Cloud Summit, Fairfax, VA, USA, 2022, pp. 91-96, doi: 10.1109/CloudSummit54781.2022.00020.

[19] E. Qumsiyeh and I. Ishaq, ""Smart Parking Navigator": An IoT Fog-Based System for Realtime Car Parking Detection," 2023 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 2023, pp. 0487-0492, doi: 10.1109/AIIoT58121.2023.10174475.

[20] M. A. Mokar, S. O. Fageeri and S. E. Fattoh, "Using Firebase Cloud Messaging to Control Mobile Applications," 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2019, pp. 1-5, doi: 10.1109/ICCCEEE46830.2019.9071008.

[21] M. A. Mokar, S. O. Fageeri and S. E. Fattoh, "Building a model to unify E-payment and access in Sudan with help of firebase cloud messaging," 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2021, pp. 1-4, doi: 10.1109/ICCCEEE49695.2021.9429645.

[23] M. Babiuch and P. FoltÝnek, "Creating a Mobile Application with the ESP32 Azure IoT Development Board Using a Cloud Platform," 2021 22nd International Carpathian Control Conference (ICCC), Velké Karlovice, Czech Republic, 2021, pp. 1-4, doi: 10.1109/ICCC51557.2021.9454607.

[22] S. K, A. X. K, D. Davis and N. Jayapandian, "Internet of Things and Cloud Computing Involvement Microsoft Azure Platform," 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 2022, pp. 603-609, doi: 10.1109/ICECAA55415.2022.9936126.

[23] C. Vuppalapati, A. Ilapakurti, K. Chillara, S. Kedari and V. Mamidi, "Automating Tiny ML Intelligent Sensors DevOPS Using Microsoft Azure," 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 2020, pp. 2375-2384, doi: 10.1109/BigData50022.2020.9377755.

[24] S. Forsström and U. Jennehag, "A performance and cost evaluation of combining OPC-UA and Microsoft Azure IoT Hub into an industrial Internet-of-Things system," 2019 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 2019, pp. 1-6, doi: 10.1109/GIOTS.2017.8016265.

[25] M. Babiuch and P. FoltÝnek, "Creating a Mobile Application with the ESP32 Azure IoT Development Board Using a Cloud Platform," 2021 22nd International Carpathian Control Conference (ICCC), Velké Karlovice, Czech Republic, 2021, pp. 1-4, doi: 10.1109/ICCC51557.2021.9454607.

[26] Ayman A. Wazwaz, Khalid M. Amin, Noura A. Semari, Tamer F. Ghanem, “Enhancing human activity recognition using features reduction in IoT edge and Azure cloud”, Decision Analytics Journal, Volume 8, 2023, 100282, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2023.100282.

[27] R. Rei, J. Metrôlho and F. Ribeiro, "We Help : Platform to find service providers," 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), Aveiro, Portugal, 2023, pp. 1-6, doi: 10.23919/CISTI58278.2023.10211509.

[28] Y. Xie, A. L. Souto, S. Fachada, D. Bonatto, M. Teratani and G. Lafruit, "Performance analysis of DIBR-based view synthesis with kinect azure," 2021 International Conference on 3D Immersion (IC3D), Brussels, Belgium, 2021, pp. 1-6, doi: 10.1109/IC3D53758.2021.9687195.

[29] Timothy L. Warner, "Deploying and Configuring Azure App Service Apps," in Microsoft Azure For Dummies, Wiley, 2020, pp.139-166.

[30] R. Győrödi, M. I. Pavel, C. Győrödi and D. Zmaranda, "Performance of OnPrem Versus Azure SQL Server: A Case Study," in IEEE Access, vol. 7, pp. 15894-15902, 2019, doi: 10.1109/ACCESS.2019.2893333.

[31] D. Dundjerski and M. Tomašević, "Automatic Database Troubleshooting of Azure SQL Databases," in IEEE Transactions on Cloud Computing, vol. 10, no. 3, pp. 1604-1619, 1 July-Sept. 2022, doi: 10.1109/TCC.2020.3007016.

[32] S. R Rizvi, B. Killough, A. Cherry and S. Gowda, "Lessons Learned and Cost Analysis of Hosting a Full Stack Open Data Cube (ODC) Application on the Amazon Web Services (AWS)," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 2018, pp. 8643-8646, doi: 10.1109/IGARSS.2018.8518084.

[33] K. Swedha and T. Dubey, "Analysis of Web Authentication Methods Using Amazon Web Services," 2019 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 2018, pp. 1-6, doi: 10.1109/ICCCNT.2018.8494054.

[34] A. Mohamed, G. Gunasegaran and D. Herath, "Cloud-based Weather Condition Monitoring System using ESP8266 and Amazon Web Services," 2023 8th International Conference on Information Technology Research (ICITR), Colombo, Sri Lanka, 2023, pp. 1-5, doi: 10.1109/ICITR61062.2023.10382785.

[35] R. Botez, V. Strautiu, I. -A. Ivanciu and V. Dobrota, "Containerized Application for IoT Devices: Comparison between balenaCloud and Amazon Web Services Approaches," 2020 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2020, pp. 1-4, doi: 10.1109/ISETC50328.2020.9301070.

[36] M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," 2019 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, Colombia, 2016, pp. 179-182, doi: 10.1109/CCGrid.2016.37.

[37] H. Puripunpinyo and M. H. Samadzadeh, "Effect of optimizing Java deployment artifacts on AWS Lambda," 2019 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 2017, pp. 438-443, doi: 10.1109/INFCOMW.2017.8116416.

[38] S. Tantiphuwanart, N. Tuaycharoen, D. Wanvarie, N. Pratanwanich and A. Suchato, "Performance Improvement on a Learning Assessment Web Application Using AWS DynamoDB as a Cache Database," 2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE), Phitsanulok, Thailand, 2023, pp. 303-308, doi: 10.1109/JCSSE58229.2023.10201973.

[39] R. K. Kodali and A. C. Sabu, "Aqua Monitoring System using AWS," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-5, doi: 10.1109/ICCCI54379.2022.9740798.

[40] Ben Piper; David Clinton, "Authentication and Authorization—AWS Identity and Access Management," in AWS Certified Solutions Architect Study Guide with 900 Practice Test Questions: Associate (SAA-C03) Exam, Wiley, 2023, pp.175-192.

[41] D. -H. Kang et al., "Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT on AWS," 2018 International Conference on Platform Technology and Service (PlatCon), Busan, Korea (South), 2017, pp. 1-5, doi: 10.1109/PlatCon.2017.7883724.