



Universidad Técnica Estatal de Quevedo

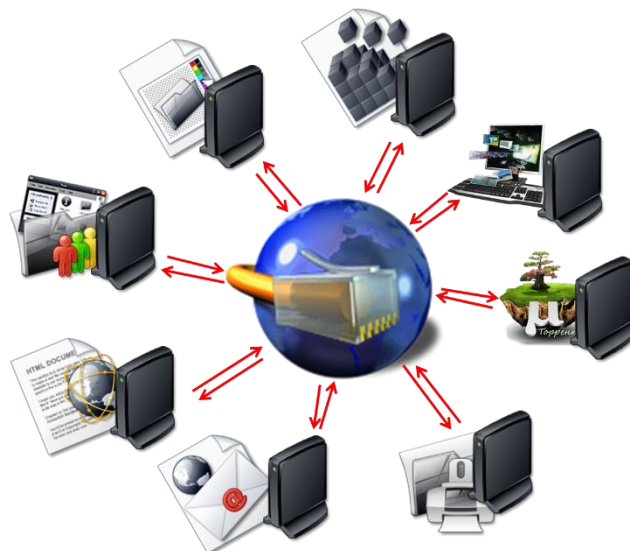
Estudiante: Cedeño Orlando, Vilcacundo Jordy, Robalino Bryan y Orrala William

Curso: Ingeniería en Software 7mo "A".

Tema: Desarrollo de Aplicación en Capas.

Asignatura: Aplicaciones Distribuidas.

Docente: Guerrero Ulloa Gleiston Cicerón.



Índice

1.	Introducción.....	5
2.	Fundamentos del desarrollo en capas.....	6
3.	Arquitectura de la aplicación en capas.....	6
4.	Firebase.....	7
4.1.	Introducción a Firebase	7
4.2.	Características Principales	7
4.2.1.	Integración completa	8
4.2.2.	Desarrollo sin servidor.....	8
4.2.3.	Autenticación de usuario.....	9
4.2.4.	Realtime Database y FireStore	10
4.2.5	Realtime database	10
4.2.6	Cloud firestore.....	10
4.3.	Firebase como plataforma de desarrollo de aplicaciones en tiempo real. 11	
4.3.1	Capacidades en Tiempo Real.....	11
4.3.2	Integración y Escalabilidad	11
4.3.3	Desarrollo Ágil y Experiencia de Usuario Mejorada	11
4.4.	Uso de Firebase Authentication para autenticación de usuarios.....	12
4.4.1	Facilidad de Integración y Amplia Gama de Métodos de Autenticación 12	
4.4.2	Seguridad y Cumplimiento	12
4.4.3	Integración con Otros Servicios de Firebase.....	13
4.4.4	Experiencia de Usuario Mejorada	13
4.5.	Firebase Realtime Database para almacenamiento y sincronización de datos.13	
4.5.1	Almacenamiento Dinámico de Datos.....	13
4.5.2	Sincronización en Tiempo Real	14

4.5.3	Desarrollo de Aplicaciones Offline.....	14
4.5.4	Seguridad y Control de Acceso	14
4.6.	Firebase Cloud Functions para la lógica de servidor sin servidor.	14
4.6.1	Eventos Disparadores	15
4.6.2	Integración y Escalabilidad Automática.....	15
4.6.3	Desarrollo Eficiente y Foco en la Experiencia del Usuario.....	15
5.	Microsoft Azure	15
5.1	Introducción a Microsoft Azure.....	15
5.2	Características Principales	16
5.3	Azure como plataforma de servicios en la nube.	16
5.4	Azure App Service para la implementación de aplicaciones web y móviles. 17	
5.5	Azure Functions para la creación de funciones sin servidor.	18
5.6	Azure SQL Database para el almacenamiento de datos.....	19
6.	Amazon Web Services	19
6.2.1	Amazon Elastic Compute Cloud (Amazon EC2)	20
6.2.2	Amazon Machine Image (Amazon AMI)	21
7.	Practica	23
7.1	Firebase.....	23
7.2	Migración de SQL SERVER a AZURE SQL	41
8.	Conclusión.....	51
9.	Referencias	52

Ilustraciones

Ilustración 1.	Pantalla de inicio de FireBase, Vilcacundo Jordy	24
Ilustración 2.	Repositorio de proyectos en firebase, Vilcacundo Jordy	24
Ilustración 3.	Asignación de nombre para el proyecto, Vilcacundo Jordy	25
Ilustración 4.	Selección de Google Analytics, Vilcacundo Jordy	25

Ilustración 5. Pantalla de espera, Vilcacundo Jordy	26
Ilustración 6. Confirmación de creación, Vilcacundo Jordy	26
Ilustración 7. Pantalla de inicio de firebase, Vilcacundo Jordy	27
Ilustración 8. Registro de nombre de app, Vilcacundo Jordy	27
Ilustración 9. SDK de firebase, Vilcacundo Jordy	28
Ilustración 10. Consola de firebase, Vilcacundo Jordy	28
Ilustración 11. Métodos de inicio de sesión, Vilcacundo Jordy	29
Ilustración 12. Consola de firebase, Vilcacundo Jordy	29
Ilustración 13. Creación de base de datos temporal, Vilcacundo Jordy.....	30
Ilustración 14. Creación de una nueva colección, Vilcacundo Jordy	30
Ilustración 15. Creación de tabla y campos, Vilcacundo Jordy	31
Ilustración 16. Base de datos Users, Vilcacundo Jordy	31
Ilustración 17. Interfaz de inicio de sesión, Vilcacundo Jordy	35
Ilustración 18. Interfaz de registro de usuarios, Vilcacundo Jordy	38
Ilustración 19. Usuarios registrados en la base de datos, Vilcacundo Jordy	38
Ilustración 20: Inicio de Sesión con Google, Vilcacundo Jordy	40
Ilustración 21. Información de usuario, Vilcacundo Jordy	40
Ilustración 22: Inicio de Sesión con credenciales, Vilcacundo Jordy	41
Ilustración 23: Sesión iniciada, Vilcacundo Jordy	41
Ilustración 24: Selección de base de datos a migrar, Cedeño Orlando.....	42
Ilustración 25: Export Data-tier Application, Cedeño Orlando.	42
Ilustración 26: Creación de archivo Bacpac, Cedeño Orlando.	43
Ilustración 27: Exportar configuración de archivo Bacpac, Cedeño Orlando.....	44
Ilustración 28: Archivo Bacpac creado, Cedeño Orlando.	45
Ilustración 29: Comprobación de no existencia de errores al crear el archivo bacpac, Cedeño Orlando.....	46
Ilustración 30: Creación de conexión en Azure Data Studio, Cedeño Orlando.	47
Ilustración 31: Configuración de Conexión, Cedeño Orlando.	47
Ilustración 32: Aceptación de Certificado, Cedeño Orlando.	48
Ilustración 33: Instalación de complemento SQL Server Dacpac, Cedeño Orlando.	48
Ilustración 34: Uso de Data-tier Application Wizard, Cedeño Orlando.....	49
Ilustración 35: Abrir base de datos del archivo bacpac, Cedeño Orlando.	49
Ilustración 36: Comprobación de base de datos migrada, Cedeño Orlando.	50

Ilustración 37: Base de datos migrada en Azure Sql, Cedeño Orlando.....	50
Ilustración 38: Tablas migradas, Cedeño Orlando.....	51
Ilustración 39: Consulta de registros de la tabla Region, Cedeño Salvatierra.....	51

1. Introducción

El proceso de desarrollo de software mediante la implementación de una arquitectura de múltiples capas representa una estrategia metodológica que organiza la construcción y el diseño de aplicaciones en segmentos claramente diferenciados [1] . Esta táctica no solo mejora la cohesión entre las distintas secciones de la aplicación, permitiendo un enfoque más sistemático y ordenado en su desarrollo, sino que también facilita la interoperabilidad y la colaboración entre equipos al usar un conjunto variado de herramientas específicas para cada capa [1]. Al adherirse a los conceptos fundamentales de la programación orientada a objetos, como la encapsulación que protege el estado interno de un objeto y la herencia que permite la creación de nuevas clases basadas en clases existentes, este enfoque promueve una estructura de software más robusta, flexible y mantenible [2].

Adicionalmente, el principio de diseño por capas trasciende el ámbito del desarrollo de software, encontrando aplicabilidad en diversas disciplinas técnicas y científicas. Un ejemplo notable es el análisis del comportamiento de fluidos a través de materiales porosos, como los textiles, donde se emplean modelos discretos basados en sistemas de celdas y conductos para simular de manera precisa la distribución de la humedad en el tejido, organizando el material en múltiples capas para replicar las interacciones complejas entre el fluido y la estructura del material [3].

En el campo de la geociencia, la aplicación de estructuras estratificadas desempeña un papel crucial en el almacenamiento subterráneo de hidrógeno, una práctica emergente en la búsqueda de soluciones sostenibles para la gestión de energía [4]. La complejidad de estas estructuras requiere el desarrollo de modelos analíticos avanzados para predecir las respuestas termo mecánicas de los pozos de almacenamiento, que a menudo se componen de varias capas de materiales con propiedades físicas distintas [5]. Estos modelos son fundamentales para asegurar la viabilidad y la seguridad de las instalaciones de almacenamiento, proyectando las

tensiones y deformaciones que podrían afectar la integridad de los pozos a lo largo del tiempo [2].

2. Fundamentos del desarrollo en capas

El desarrollo en capas es un enfoque fundamental en las aplicaciones distribuidas. Implica dividir la aplicación en diferentes capas, cada una responsable de un aspecto específico del sistema. Esto permite el diseño modular, el desarrollo independiente y una gestión más fácil de las responsabilidades entre los miembros del equipo [6]. La arquitectura en capas proporciona una descripción clara de las actividades y dirige a la fuerza laboral, lo que permite una colaboración y coordinación eficientes [7]. Además, los métodos de diseño en capas y el desarrollo colaborativo utilizando diferentes herramientas contribuyen a la optimización integrada en el desarrollo de sistemas de grandes redes [8]. El enfoque por capas también facilita la especificación, diseño y análisis de cada capa de forma independiente, asegurando las propiedades del sistema completo. Esto es particularmente útil en entornos adversarios, como la seguridad y los protocolos criptográficos, donde la especificación y el análisis precisos son cruciales [9].

3. Arquitectura de la aplicación en capas

Separa una aplicación en distintas capas lógicas o componentes, distribuidos en diferentes niveles o niveles de abstracción, con el objetivo de mejorar el modularidad, la escalabilidad y la mantenibilidad del sistema [10].

En este contexto, las capas representan diferentes responsabilidades funcionales y pueden incluir la presentación (frontend), la lógica de negocio (backend) y el almacenamiento de datos (base de datos) [10]. Cada capa se comunica con las capas adyacentes a través de interfaces bien definidas, lo que permite una mayor flexibilidad y facilita la evolución independiente de cada componente [9].

Las aplicaciones distribuidas son aquellas en las que los componentes o módulos de la aplicación se ejecutan en múltiples dispositivos o servidores conectados a través de una red [9]. Esto puede incluir arquitecturas cliente-servidor, arquitecturas de microservicios o incluso sistemas distribuidos a gran escala en la nube [9][11].

Al combinar la arquitectura en capas con la distribución de la aplicación, se pueden obtener una serie de beneficios, como la capacidad de escalar y distribuir

componentes individualmente para satisfacer la demanda, la posibilidad de implementar nuevas funcionalidades de manera modular sin afectar a otras partes del sistema, y la mejora en la gestión de la complejidad al dividir la aplicación en unidades más pequeñas y manejables [11].

4. Firebase

4.1. Introducción a Firebase

Firebase es una plataforma integral para el desarrollo de aplicaciones web y móviles que fue lanzada originalmente en 2011 y posteriormente adquirida por Google en 2014. Se ha convertido en una solución popular entre los desarrolladores debido a su amplio conjunto de herramientas y servicios que facilitan el desarrollo rápido de aplicaciones, permitiendo la implementación de funcionalidades complejas sin la necesidad de gestionar la infraestructura subyacente [12]. Firebase ofrece desde soluciones de backend como bases de datos en tiempo real y autenticación de usuarios hasta herramientas de análisis y crecimiento de aplicaciones, apoyando a los desarrolladores a través de todo el ciclo de vida de la aplicación [13].

Firebase se destaca por su capacidad para acelerar el proceso de desarrollo de aplicaciones, proporcionando una serie de servicios que incluyen, pero no se limitan a, autenticación de usuarios, almacenamiento en la nube, notificaciones en tiempo real, análisis de uso, y la ejecución de código backend sin servidores a través de Cloud Functions [14]. La integración de estos servicios en una plataforma unificada permite a los desarrolladores concentrarse en la experiencia del usuario y la lógica de la aplicación, reduciendo el tiempo y el esfuerzo necesarios para implementar infraestructuras complejas [13].

4.2. Características Principales

Firebase se ha consolidado como una plataforma líder para el desarrollo de aplicaciones web y móviles, ofreciendo un conjunto de herramientas y servicios integrados que abordan los desafíos comunes en el desarrollo de aplicaciones. Las características principales de Firebase lo hacen destacar por su eficiencia, facilidad de uso y la capacidad de escalar aplicaciones rápidamente [15].

4.2.1. Integración completa

La integración completa de Firebase se destaca como una de sus características más valiosas, ofreciendo a los desarrolladores una solución todo en uno para el desarrollo y la gestión de aplicaciones móviles y web. Esta característica permite a los equipos de desarrollo utilizar un conjunto cohesivo de herramientas y servicios que están profundamente integrados entre sí, facilitando una experiencia de desarrollo sin fisuras [16]. Desde la autenticación de usuarios hasta el almacenamiento en la nube, pasando por bases de datos en tiempo real y análisis de aplicaciones, Firebase proporciona una plataforma unificada que elimina la necesidad de juntar servicios dispares de múltiples proveedores [17]. Esta integración completa no solo simplifica el proceso de desarrollo, sino que también reduce los tiempos de implementación y permite a los desarrolladores centrarse más en innovar y mejorar la experiencia del usuario en lugar de lidiar con la complejidad técnica [18].

Además, la integración completa de Firebase facilita la colaboración entre los diferentes miembros del equipo de desarrollo, ya que todos pueden trabajar dentro del mismo ecosistema de herramientas. Esto mejora la eficiencia y la cohesión del proyecto, ya que las actualizaciones, los cambios y las nuevas implementaciones se pueden realizar y monitorear en tiempo real, asegurando que todos los componentes de la aplicación trabajen armónicamente [19]. Esta característica de Firebase se traduce en un ciclo de desarrollo más ágil y una capacidad de respuesta más rápida a las necesidades del mercado o a los requerimientos de los usuarios, permitiendo a las empresas lanzar aplicaciones robustas y de alta calidad con una inversión de tiempo y recursos significativamente menor [20].

4.2.2. Desarrollo sin servidor

El desarrollo sin servidor es otra característica destacada de Firebase, que transforma radicalmente la forma en que los desarrolladores construyen y escalan aplicaciones. A través de Firebase Cloud Functions, la plataforma permite ejecutar código backend en respuesta a eventos desencadenados por servicios de Firebase o solicitudes HTTPS, sin la necesidad de administrar o escalar una infraestructura de servidores propia [20].

Firebase Cloud Functions se integra a la perfección con otros servicios de Firebase y Google Cloud, proporcionando un ecosistema rico y flexible para el desarrollo de aplicaciones. Por ejemplo, los desarrolladores pueden crear funciones que se activan en respuesta a cambios en Firebase Realtime Database, Firestore, eventos de autenticación, notificaciones push, y mucho más [21]. Esto habilita un modelo de desarrollo reactivo, donde el código backend responde automáticamente a las interacciones de los usuarios en tiempo real, mejorando la eficiencia y la interactividad de las aplicaciones [22].

4.2.3. Autenticación de usuario

La autenticación de usuarios es un pilar fundamental dentro de las características que ofrece Firebase, proporcionando un sistema completo y seguro para la gestión de identidades y el acceso a las aplicaciones [23]. Firebase Authentication facilita la implementación de procesos de inicio de sesión y registro en las aplicaciones, soportando una amplia gama de métodos de autenticación, incluyendo correo electrónico y contraseña, autenticación telefónica, y logins mediante proveedores de identidad externos como Google, Facebook, Twitter, y GitHub [24]. Esta diversidad de opciones permite a los desarrolladores ofrecer a sus usuarios la flexibilidad de elegir su método preferido de autenticación, mejorando la experiencia de usuario y fomentando una mayor tasa de adopción.

Lo que hace especialmente valioso a Firebase Authentication es su integración nativa con otros servicios de Firebase, lo que permite a los desarrolladores construir sistemas de seguridad robustos con mínima configuración. Por ejemplo, las reglas de seguridad de Firebase Realtime Database y Cloud Firestore pueden configurarse para restringir el acceso a los datos basándose en la identidad del usuario, asegurando que los usuarios solo puedan acceder a la información que les corresponde [25]. Además, Firebase Authentication maneja aspectos complejos de la autenticación, como la verificación de correo electrónico, la recuperación de contraseñas y la gestión de sesiones, liberando a los desarrolladores de implementar estas funcionalidades desde cero y garantizando que las prácticas de seguridad estén alineadas con las mejores prácticas de la industria [26]. Esto no solo agiliza el desarrollo de aplicaciones seguras, sino que también proporciona tranquilidad tanto

a desarrolladores como a usuarios finales en cuanto a la protección de sus datos e identidad.

4.2.4. Realtime Database y FireStore

Firebase ofrece dos poderosas soluciones de bases de datos diseñadas para facilitar el desarrollo de aplicaciones interactivas y en tiempo real: Firebase Realtime Database y Cloud Firestore. Ambas están diseñadas para ayudar a los desarrolladores a almacenar y sincronizar datos entre los usuarios en tiempo real, pero cada una tiene sus propias características únicas y casos de uso ideales [26].

4.2.5 Realtime database

Firebase Realtime Database es una base de datos NoSQL alojada en la nube que permite a los desarrolladores almacenar y sincronizar datos entre sus usuarios en tiempo real. Cada vez que los datos cambian, todos los dispositivos conectados reciben automáticamente esa actualización en cuestión de milisegundos, proporcionando una experiencia de usuario fluida y dinámica [27]. Esto es particularmente útil para aplicaciones que requieren una alta interactividad entre los usuarios, como juegos en tiempo real, aplicaciones de chat, y colaboraciones en vivo. La Realtime Database también ofrece una API sencilla y flexible, permitiendo a los desarrolladores construir aplicaciones rápidamente sin tener que preocuparse por los detalles de sincronización de datos o la gestión de conexiones de red [28].

4.2.6 Cloud firestore

Cloud Firestore es la nueva generación de Firebase Realtime Database, con una serie de mejoras y características adicionales. Es una base de datos NoSQL también alojada en la nube que ofrece sincronización de datos en tiempo real y capacidades offline, junto con una modelación de datos más rica y consultas más potentes en comparación con Firebase Realtime Database [12]. Firestore está diseñado para escalar mejor que su predecesor, facilitando el desarrollo de aplicaciones más grandes y complejas. Ofrece estructuras de datos basadas en colecciones y documentos, que se pueden utilizar para almacenar y modelar datos de forma más intuitiva. Además, Firestore mejora la experiencia de desarrollo con características como transacciones y consultas complejas, índices automáticos y una consola de administración más robusta [29].

4.3. Firebase como plataforma de desarrollo de aplicaciones en tiempo real.

Firestore se ha establecido como una plataforma líder para el desarrollo de aplicaciones en tiempo real, proporcionando a los desarrolladores un conjunto de herramientas potentes y fáciles de usar que permiten la creación de experiencias de usuario interactivas y dinámicas [30]. Al centrarse en la sincronización de datos en tiempo real y la comunicación instantánea, Firestore ofrece capacidades que son esenciales para aplicaciones modernas que requieren compartir y actualizar información en el momento en que ocurre, como aplicaciones de mensajería, juegos en línea, y aplicaciones colaborativas [31].

4.3.1 Capacidades en Tiempo Real

La base de Firestore para aplicaciones en tiempo real se asienta principalmente en dos de sus servicios: Firestore Realtime Database y Cloud Firestore. Estas bases de datos NoSQL alojadas en la nube permiten a los desarrolladores almacenar y sincronizar datos entre los usuarios en tiempo real [32]. Cualquier cambio en los datos se propaga instantáneamente a todos los dispositivos conectados, sin necesidad de refrescar o realizar consultas adicionales. Esto asegura que los usuarios siempre tengan acceso a la información más actualizada, mejorando significativamente la interactividad y la experiencia del usuario [33].

4.3.2 Integración y Escalabilidad

Firestore no solo destaca por su capacidad de sincronización en tiempo real, sino también por su integración fluida con otros servicios de Firestore y Google Cloud, lo que permite a los desarrolladores construir, escalar y mejorar sus aplicaciones con facilidad [34]. Por ejemplo, la integración con Firestore Authentication ofrece soluciones robustas para la gestión de usuarios, mientras que Firestore Cloud Functions permite ejecutar código backend en respuesta a eventos en tiempo real, sin la necesidad de administrar servidores. Esta sinergia entre servicios facilita la creación de aplicaciones complejas y ricas en características, al tiempo que mantiene la gestión de infraestructura al mínimo [35].

4.3.3 Desarrollo Ágil y Experiencia de Usuario Mejorada

Firestore proporciona a los desarrolladores las herramientas necesarias para construir aplicaciones rápidas y reactivas que responden a las acciones de los usuarios en tiempo real [35], [36]. Esto no solo mejora la experiencia del usuario,

sino que también acelera el ciclo de desarrollo, ya que Firebase ofrece una serie de servicios preconstruidos que reducen la cantidad de código necesario y eliminan la necesidad de construir soluciones personalizadas para problemas comunes. Así, los equipos pueden enfocarse en innovar y perfeccionar la experiencia del usuario en lugar de lidiar con complejidades técnicas [37].

4.4. Uso de Firebase Authentication para autenticación de usuarios.

Firebase Authentication se destaca como una solución integral para la autenticación de usuarios en aplicaciones desarrolladas con Firebase. Este servicio simplifica el proceso de integración de sistemas de autenticación seguros y personalizables, permitiendo a los desarrolladores centrarse en la lógica de la aplicación y la experiencia del usuario, mientras Firebase maneja la complejidad asociada con la autenticación de usuarios [25].

4.4.1 Facilidad de Integración y Amplia Gama de Métodos de Autenticación

Firebase Authentication proporciona una API flexible y fácil de usar que permite integrar rápidamente la autenticación en aplicaciones web y móviles. Soporta una amplia variedad de métodos de autenticación, incluyendo correo electrónico y contraseña, autenticación telefónica, y logins a través de proveedores de identidad populares como Google, Facebook, Twitter, y GitHub [38]. Esta diversidad garantiza que los desarrolladores puedan ofrecer a sus usuarios múltiples opciones para registrarse e iniciar sesión, mejorando la accesibilidad y la experiencia del usuario en general [39].

4.4.2 Seguridad y Cumplimiento

Firebase Authentication está diseñado con un enfoque en la seguridad. Implementa prácticas de seguridad líderes en la industria para proteger la información de los usuarios y asegurar las transacciones de autenticación. Además, Firebase Authentication facilita el cumplimiento de normativas globales sobre privacidad y seguridad de datos, como el GDPR en Europa y la CCPA en California, proporcionando características como la eliminación de cuentas de usuario y el acceso a los datos personales a pedido del usuario [40].

4.4.3 Integración con Otros Servicios de Firebase

Una de las mayores ventajas de Firebase Authentication es su integración nativa con otros servicios de Firebase, como Firebase Realtime Database, Cloud Firestore, y Firebase Storage. Esto permite a los desarrolladores implementar reglas de seguridad basadas en la identidad del usuario, restringiendo el acceso a los datos y recursos de la aplicación según las credenciales de autenticación del usuario [40]. Por ejemplo, se pueden configurar reglas en Firebase Realtime Database para que solo los usuarios autenticados puedan leer o escribir en ciertas áreas de la base de datos.

4.4.4 Experiencia de Usuario Mejorada

Firebase Authentication no solo beneficia a los desarrolladores por su facilidad de implementación y seguridad, sino que también mejora la experiencia del usuario final. Ofrece características como la verificación de correo electrónico, el restablecimiento de contraseñas y la gestión de sesiones, lo que contribuye a crear un sistema de autenticación de usuarios fluido y confiable [41].

4.5. Firebase Realtime Database para almacenamiento y sincronización de datos. Firebase Realtime Database es un servicio de base de datos NoSQL alojado en la nube que permite a los desarrolladores almacenar y sincronizar datos entre sus usuarios en tiempo real [42]. Esta potente herramienta de Firebase transforma la manera en que se construyen las aplicaciones interactivas, permitiendo una comunicación fluida y en tiempo real entre los usuarios y la aplicación. A continuación, se detallan aspectos clave de Firebase Realtime Database para el almacenamiento y la sincronización de datos [43].

4.5.1 Almacenamiento Dinámico de Datos

Firebase Realtime Database proporciona un almacenamiento de datos flexible y dinámico, que soporta estructuras de datos de tipo JSON. Esto significa que los desarrolladores pueden almacenar fácilmente datos estructurados en un formato que es tanto fácil de leer como de escribir en tiempo real desde la aplicación cliente [44]. La capacidad de modelar datos complejos sin necesidad de establecer esquemas fijos previos ofrece a los desarrolladores la flexibilidad para adaptarse rápidamente a las necesidades cambiantes de su aplicación [45].

4.5.2 Sincronización en Tiempo Real

Una de las características más destacadas de Firebase Realtime Database es su capacidad para sincronizar datos en tiempo real a través de todos los clientes conectados [45]. Cuando los datos cambian, estos se transmiten instantáneamente a todos los dispositivos, ya sean web o móviles, que estén escuchando esos cambios. Esto es ideal para aplicaciones que requieren una alta interactividad entre usuarios, como juegos multijugador, aplicaciones de chat, o plataformas de colaboración en tiempo real [46].

4.5.3 Desarrollo de Aplicaciones Offline

Firebase Realtime Database también soporta capacidades offline, permitiendo a las aplicaciones seguir funcionando de manera eficiente incluso sin una conexión a internet [46]. Los cambios realizados por los usuarios en modo offline se almacenan localmente y se sincronizan con la base de datos en la nube una vez que el dispositivo recupera la conexión a internet. Esta característica asegura una experiencia de usuario ininterrumpida y mejora la robustez de las aplicaciones frente a condiciones de red fluctuantes [47].

4.5.4 Seguridad y Control de Acceso

La seguridad de los datos es una prioridad en Firebase Realtime Database, que ofrece reglas de seguridad configurables para controlar el acceso a los datos almacenados. Los desarrolladores pueden definir reglas basadas en la autenticación del usuario, la validación de datos y las condiciones de acceso para asegurar que solo los usuarios autorizados puedan leer o escribir en la base de datos [48].

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

Firebase Cloud Functions representa una parte esencial del ecosistema Firebase, ofreciendo una solución poderosa y flexible para ejecutar lógica de backend en un entorno sin servidor [49]. Esta característica permite a los desarrolladores escribir funciones individuales que se ejecutan en respuesta a eventos específicos en Firebase y otros servicios de Google Cloud, eliminando la necesidad de administrar o escalar servidores propios. A continuación, se detallan aspectos clave de Firebase Cloud Functions y cómo facilita la implementación de lógica de servidor sin servidor [50].

4.6.1 Eventos Disparadores

Una de las características distintivas de Firebase Cloud Functions es su capacidad para responder automáticamente a eventos generados en Firebase y Google Cloud. Estos eventos pueden incluir cambios en Firebase Realtime Database, Firestore, autenticaciones de usuario, cargas de archivos a Firebase Storage, y mensajes enviados a través de Firebase Cloud Messaging, entre otros [50]. Al vincular funciones a estos eventos, los desarrolladores pueden automatizar tareas, como enviar notificaciones push, realizar operaciones de mantenimiento de datos, y mucho más, sin intervención manual [49].

4.6.2 Integración y Escalabilidad Automática

Firebase Cloud Functions se integra perfectamente con otros servicios de Firebase y Google Cloud, permitiendo a los desarrolladores construir aplicaciones complejas y altamente interactivas [50]. La plataforma maneja automáticamente la escalabilidad de las funciones, asegurando que tengan los recursos necesarios para ejecutarse eficientemente bajo cualquier carga de trabajo. Esto significa que las aplicaciones pueden escalar de cero a millones de usuarios sin requerir ajustes en la infraestructura [49].

4.6.3 Desarrollo Eficiente y Foco en la Experiencia del Usuario

Al mover la lógica del servidor a Cloud Functions, los desarrolladores pueden centrarse más en el diseño y la experiencia del usuario de la aplicación, en lugar de preocuparse por la gestión de la infraestructura del servidor. Esto no solo reduce el tiempo y el costo asociados con el desarrollo y mantenimiento de servidores, sino que también permite una iteración más rápida y la implementación de nuevas características sin afectar la experiencia del usuario [49].

5. Microsoft Azure

5.1 Introducción a Microsoft Azure

La evolución de Microsoft Azure hacia una "nube inteligente" que permite tecnologías avanzadas como la inteligencia artificial, el Internet de las cosas y el aprendizaje automático. El objetivo es brindar a los lectores una base sólida en todos los aspectos importantes de Azure, desde conceptos básicos hasta servicios avanzados [51].

Es una sólida plataforma en la nube que ofrece una amplia gama de productos y servicios. Azure facilita el desarrollo y la escalabilidad de aplicaciones en la nube, atendiendo tanto a las aplicaciones nuevas como a las existentes y preocupaciones comunes de seguridad en la computación en la nube, como las vulnerabilidades y las amenazas, destacando su impacto potencial en las empresas [51].

Esta plataforma es importante en el mundo de la tecnología de la información. Azure está en el centro de las decisiones tecnológicas actuales, particularmente cuando se implementan sistemas de nube en entornos privados, públicos o híbridos, el concepto de computación en la nube existe desde hace décadas, aunque el concepto en sí se ha vuelto más popular recientemente [52].

5.2 Características Principales

Microsoft Azure proporciona una plataforma integral para crear, administrar y escalar soluciones de análisis de la nube que brindan seguridad, escalabilidad y herramientas específicas para análisis de datos.

Azure proporciona herramientas para la protección y auditoría de datos según los estándares empresariales, además, proporciona análisis interactivos, de streaming y de gran volumen para optimizar las soluciones de big data permite la implementación de la gobernanza de datos para garantizar la calidad, integridad y seguridad de los datos [51].

En general, Microsoft Azure incluye ser plataforma líder en la nube, servicios y herramientas, enfoque en la seguridad integral y la gestión eficaz de costos, y capacidad para satisfacer necesidades técnicas y no técnicas. Los usuarios técnicos que quieran aprender y explorar el campo de la computación en la nube, especialmente Azure [52].

5.3 Azure como plataforma de servicios en la nube.

La tecnología de la nube transforma los servicios de datos en recursos fácilmente disponibles e introduce complejidades y oportunidades. Esto resalta la dificultad de configurar y optimizar estos servicios para maximizar su valor, además del aprendizaje automático y enfoques basados en datos para automatizar aspectos de los servicios de datos para crear servicios autónomos. Es decir, sobre la creación de estos servicios en Azure [53].

Microsoft Azure en el mercado de servicios en la nube destaca su capacidad para ampliar su negocio y ofrecer una amplia gama de servicios. Se exploran en detalle los servicios clave de Azure, incluidos la inteligencia artificial, el análisis de datos y la IoT, así como las capacidades específicas de cada servicio, como Azure Cognitive Search y Face Services. Además, en términos de capacidades informáticas, Azure se compara con otros proveedores como Google Compute Engine y Amazon Web Services, lo que proporciona una visión general amplia del entorno de servicios en la nube [54].

La importancia de la calidad en la adopción de la nube y cómo afecta la elección de plataformas de inicio. Para evaluar la calidad del uso de las plataformas de nube pública, se realizaron tres experimentos comparando AWS y Azure. AWS obtuvo mejores resultados en términos de eficiencia y se consideró más preferible en términos de interactividad y facilidad de uso [52].

5.4 Azure App Service para la implementación de aplicaciones web y móviles.

La tecnología en la nube facilita la migración de aplicaciones móviles, ahorrando costes de TI y aumentando la productividad de las empresas, a medida que aumenta el uso de Internet móvil, las empresas se centran en desarrollar aplicaciones flexibles y escalables para interactuar con los consumidores [55].

Azure Application Services simplifica la implementación y configuración de aplicaciones web en la nube, eliminando la necesidad de administrar la infraestructura de máquinas virtuales, permite extensiones hiperconvergentes y una integración perfecta con otros servicios de Azure, el servicio aloja aplicaciones web HTTP estáticas y dinámicas y proporciona herramientas como API, lógica de aplicación y funciones para facilitar el desarrollo de aplicaciones sin preocuparse por la infraestructura subyacente [55].

Los servicios de Azure son esenciales para desarrollar e implementar aplicaciones móviles, ofrecen una amplia gama de herramientas de desarrollo backend, incluidos Azure Active Directory y Azure Functions, así como opciones de almacenamiento como Cosmos DB, Además, proporcionan una integración perfecta con la nube, promoviendo la escalabilidad y el alto rendimiento de las aplicaciones móviles. Con opciones de implementación como PaaS y SaaS [55][56].

5.5 Azure Functions para la creación de funciones sin servidor.

El surgimiento de la informática sin servidor para aplicaciones escalables basadas en eventos y su enfoque en funciones sin estado basadas en eventos. Aunque la informática sin servidor ahorra recursos, todavía enfrenta desafíos en varias características. Se analizan las limitaciones de incorporar esta combinación en aplicaciones sin servidor y se proponen extensiones para abordar este problema, la computación sin servidor se puede mejorar mediante soluciones que admitan la composición secuencial funcional [56].

Una plataforma de computación sin servidor permite a los desarrolladores alojar aplicaciones escalables automáticamente, estas aplicaciones se crean y ejecutan en contenedores, pero se señala que esto puede generar sobrecarga, especialmente para aplicaciones sensibles a la latencia o en entornos avanzados. Se propone WebAssembly como alternativa para ejecutar aplicaciones sin servidor, ofreciendo rendimiento similar a los contenedores pero con tiempos de inicio más rápidos y menor consumo de recursos [57].

Este es un enfoque de capacidades de servidor de múltiples nubes (FaaS) que organiza FaaS desde múltiples proveedores de nube para cumplir con los requisitos de las aplicaciones. Un ejemplo es que se integra un planificador jerárquico y un solucionador de satisfacción de restricciones, se analizan técnicas de programación, solucionadores de satisfacción de restricciones y coordinación del tiempo de ejecución de funciones seleccionadas y se puede utilizar funciones de la nube para probar el método [58].

Una investigación del problema del "arranque en frío" en plataformas informáticas sin servidor. Las latencias de inicio en frío y en caliente de las funciones sin servidor y lanzamos múltiples operaciones al mismo tiempo, esto se refiere a la configuración de funciones sin servidor y a cómo los proveedores de plataformas administran los recursos para evitar costos innecesarios [59].

Este ejemplo es sobre el FaaSBatch un marco para mejorar la eficiencia de la informática sin servidor al reducir la latencia de las llamadas y optimizar la utilización de recursos. FaaSBatch agrupa solicitudes de funciones simultáneas, las agrupa en un solo contenedor y reutiliza recursos redundantes para reducir el

tiempo de ejecución y el consumo de recursos, FaaSBatch reduce significativamente la latencia de las llamadas y el consumo de recursos en comparación con otros programadores de alto nivel [58], [59].

5.6 Azure SQL Database para el almacenamiento de datos.

Azure SQL Database y la próxima versión de SQL Server introducen un nuevo mecanismo de recuperación de bases de datos que combina ARIES con control de simultaneidad de múltiples versiones. La recuperación continua independientemente del tamaño de la transacción y el registro de transacciones se puede truncar continuamente incluso para transacciones de larga duración, es fundamental para los servicios de bases de datos basados en la nube debido al tamaño cada vez mayor de las bases de datos [60] .

El desarrollo de un servicio de indexación automática para Microsoft Azure SQL Database, un servicio de base de datos relacional. El servicio automatiza el proceso de administración de índices, genera sugerencias de índices y las aplica automáticamente a una gran cantidad de bases de datos en Azure SQL Database. Mejora significativamente el rendimiento de cientos de miles de bases de datos, demostrando la practicidad y eficiencia del servicio en entornos de producción [61].

Ejemplo de aplicación de Microsoft Purview es un servicio de gobernanza de datos que unifica la gestión de datos de una organización desde una sola plataforma. Automatiza el descubrimiento, clasificación y aplicación de políticas de seguridad en datos estructurados y no estructurados, incluida la integración con Office 365 y Azure SQL Database para una gobernanza centralizada [60], [61].

La evolución de los servicios de datos en la nube, señalando la facilidad de acceso pero también los desafíos en su configuración y gestión. Se propone el uso de enfoques basados en ciencia de datos y aprendizaje automático para automatizar estos servicios y crear servicios de datos autónomos [62].

6. Amazon Web Services

6.1 Introducción a Amazon Web Services (AWS).

Amazon web services (AWS) inicio la oferta de servicios de infraestructura de tecnologías de la información a empresas a través de servicios web, ahora

comúnmente conocidos como computación en la nube [63]. La ventaja clave de la computación en la nube radica en la posibilidad de sustituir los costos iniciales de infraestructura con gastos variables más bajos que se ajustan al tamaño de su empresa. Gracias a la nube, las empresas ya no necesitan anticipar y adquirir servidores y otros componentes de infraestructura de TI con semanas o meses de antelación. En su lugar, pueden iniciar cientos o miles de servidores en unos minutos, permitiendo una entrega de resultados más rápida [63].

AWS ofrece una amplia gama de herramientas y servicios diseñados para facilitar la creación de entornos de computación en la nube. Estos servicios están adaptados para atender las necesidades de empresas de diversos tamaños, ya sea que busquen desarrollar un sitio web de marketing, almacenamiento o comercio electrónico. Esta versatilidad motiva a los clientes a expandirse e innovar de acuerdo con sus metas institucionales, alentándolos a invertir en la nube [64].

6.2 AWS como plataforma de servicios en la nube.

Amazon ofrece una amplia gama de productos y servicios que se adaptan a las necesidades del usuario, distribuidos en categorías como cómputo, almacenamiento, bases de datos, redes, entrega de contenido, herramientas para desarrolladores, entre otras. Este enfoque permite brindar apoyo efectivo a los administradores de TI, profesionales de sistemas y desarrolladores para gestionar y monitorear de manera sencilla sus recursos de infraestructura híbrida [65].

6.2.1 Amazon Elastic Compute Cloud (Amazon EC2)

es un servicio basado en la web que opera en un entorno de computación en la nube seguro y escalable. En términos sencillos, se puede describir como una "máquina virtual" en la nube, brindando a los usuarios la capacidad de crear entornos de desarrollo, pruebas y gestión de aplicaciones [66].

En el marco de Amazon EC2, es posible emplear múltiples servidores virtuales según sea necesario, en cuestión de minutos. Esta flexibilidad permite adaptarse de manera dinámica a las fluctuaciones en el tráfico, escalando hacia arriba o hacia abajo según las demandas específicas. Este enfoque elimina la necesidad de realizar inversiones en hardware, ya que los usuarios solo pagan por la capacidad que realmente utilizan [66].

6.2.2 Amazon Machine Image (Amazon AMI)

El sistema de archivos de Amazon AMI se encuentra cifrado, comprimido y dividido en secciones que AWS carga y almacena con una designación que incluye nombre, versión, tipo de arquitectura, identificación del kernel y clave de descifrado. Estas imágenes pueden considerarse como plantillas que incorporan la configuración del sistema operativo, definiendo así el entorno que utilizará el usuario. Esta característica posibilita la creación rápida de múltiples instancias cuando es necesario, evitando la necesidad de configurar cada instancia de manera individual, simplificando el proceso y reduciendo el tiempo dedicado a la duplicación de información aplicaciones [66].

6.3 AWS Lambda para la ejecución de código sin servidor.

Las arquitecturas de microservicios proponen una solución para escalar eficientemente los recursos informáticos y resolver muchos otros problemas presentes en las arquitecturas monolíticas. Sin embargo, las arquitecturas de microservicios también enfrentan otros desafíos, como el esfuerzo necesario para implementar cada microservicio, así como para escalarlos y operarlos en infraestructuras en la nube [67]. Para abordar estas preocupaciones, servicios como AWS Lambda permiten implementar arquitecturas de microservicios sin la necesidad de gestionar servidores. Así, facilita la creación de funciones (es decir, microservicios) que pueden ser implementadas fácilmente y escaladas automáticamente, y también ayuda a reducir los costos de infraestructura y operación [67].

6.4 AWS DynamoDB para el almacenamiento de datos NoSQL.

La base de datos NoSQL Amazon DynamoDB fue presentada a mediados de enero de 2012. Una característica clave de DynamoDB que la hace tan atractiva es su enfoque sin servidor, que permite olvidarse de los problemas de escalabilidad de la base de datos, el mantenimiento y la configuración del servidor, y otras incomodidades asociadas con las bases de datos tradicionales [68].

DynamoDB opera bajo el concepto de "capacidad provisionada" (provisioned throughput). Al crear una tabla DynamoDB, el usuario solo necesita especificar la capacidad de lectura y escritura requerida [68]. En segundo plano, todo se configura

para satisfacer las necesidades del usuario, manteniendo una baja latencia en milisegundos. Más adelante, a medida que las necesidades del usuario crecen, pueden aumentar la capacidad asignada (o disminuirla, si es necesario). Este cambio se puede realizar en línea, sin tiempo de inactividad y sin afectar la capacidad total. En otras palabras, el usuario tiene la capacidad de ajustar la escala incluso cuando la base de datos está procesando consultas [68].

6.5 AWS Cognito para la autenticación y autorización de usuarios.

Amazon Cognito es un servicio que ofrece autenticación, autorización y gestión de usuarios para aplicaciones web y móviles. Con Cognito, los usuarios pueden optar por autenticarse con un nombre de usuario y contraseña o a través de proveedores de identidad social como Amazon, Google o Facebook. Al mismo tiempo, las empresas pueden almacenar datos localmente en los dispositivos del usuario, y la aplicación puede funcionar normalmente incluso si el dispositivo está sin conexión. Luego, las empresas pueden sincronizar datos entre los dispositivos de los usuarios para mantener consistente su experiencia de aplicación, sin importar el dispositivo que utilicen [69].

Dos componentes principales de Amazon Cognito son los grupos de usuarios (user pools) y los grupos de identidades (identity pools). En resumen, el grupo de usuarios es un directorio de usuarios que proporciona opciones de registro e inicio de sesión para los usuarios de la aplicación. El grupo de identidades puede proporcionar credenciales temporales para conceder acceso a otros servicios de AWS a los usuarios de la aplicación. El uso de estos dos grupos de manera independiente o conjunta puede satisfacer diferentes escenarios. Por ejemplo, una solución común para utilizar ambos grupos juntos es permitir que el usuario inicie sesión con el grupo de usuarios y luego acceda a los servicios de AWS utilizando el grupo de identidades [69], [70].

Además, es necesario describir más información sobre el token en sí. El token del grupo de usuarios, también conocido como JSON Web Token (JWT), consta de tres partes: encabezado, carga útil y firma. Encabezado: contiene el ID de clave y el algoritmo, Carga útil: contiene las afirmaciones sobre el usuario autenticado, como el nombre, el correo electrónico y el uso del token, Firma: se calcula en función del

encabezado y la carga útil. Cada autenticación exitosa devolverá tres tipos de tokens: token de ID, token de acceso y token de actualización [70]. Como indica el nombre del token, los tokens de ID involucran principalmente algunos casos de uso relacionados con la identidad del usuario autenticado. El token de acceso se basa en el token de ID y agrega el alcance de los derechos de acceso para diferentes grupos de usuarios. En cuanto al token de actualización, se utiliza principalmente para recuperar los nuevos tokens de ID y acceso [71].

7. Practica

7.1 Firebase

Problema: Gestión de Acceso y Control de Usuarios en Aplicaciones Web

Descripción: La gestión de acceso y control de usuarios en aplicaciones web es esencial para garantizar la seguridad de los datos del usuario y prevenir el acceso no autorizado. Sin embargo, implementar esta funcionalidad de manera eficiente y segura puede ser un desafío. La falta de un sistema adecuado puede resultar en vulnerabilidades de seguridad y una mala experiencia del usuario. Para abordar este problema, se propone desarrollar un sistema de autenticación y autorización utilizando Firebase Authentication y Cloud Firestore. Esta solución proporcionará una forma segura y escalable para que los usuarios se registren, inicien sesión y accedan a recursos autorizados. Firebase Authentication garantizará la autenticación segura de los usuarios, mientras que Cloud Firestore se utilizará para almacenar y gestionar datos de usuario, roles y permisos de acceso. La integración de estas tecnologías permitirá una gestión eficiente y confiable del acceso y control de usuarios en aplicaciones web.

Paso a paso

1. Creación de proyecto en FireBase

Para la creación de un nuevo proyecto en FireBase, nos dirigimos a [FireBase](#) daremos clic en comienza ahora.

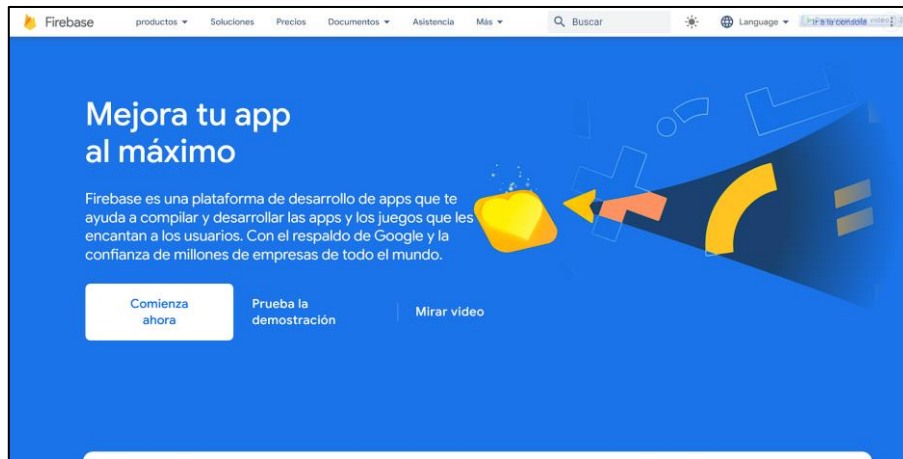


Ilustración 1. Pantalla de inicio de FireBase, Vilcacundo Jordy.

Una vez que damos clic en Comienza ahora, nos solicitara iniciar sesión, en caso de tener una sesión iniciada, nos enviara directo a nuestro repositorio de firebase, donde daremos clic en Agregar proyecto.

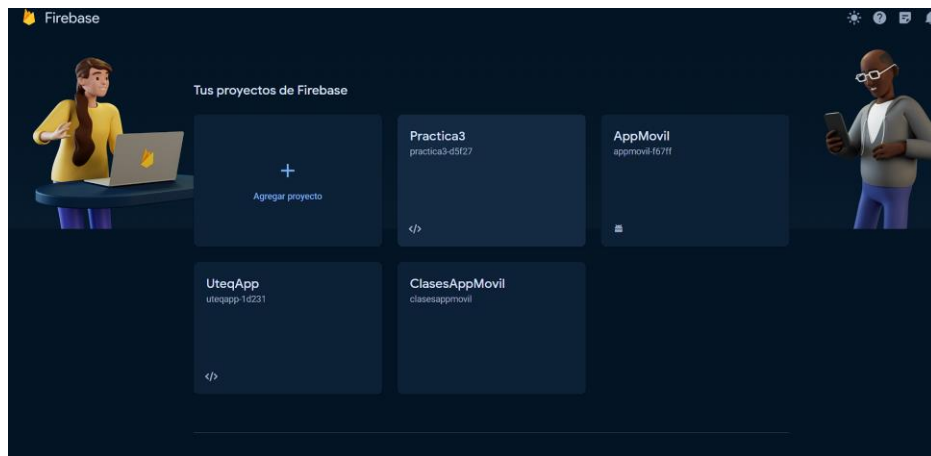


Ilustración 2. Repositorio de proyectos en firebase, Vilcacundo Jordy.

Una vez que damos clic *en agregar proyecto, le daremos un nombre a ese proyecto.

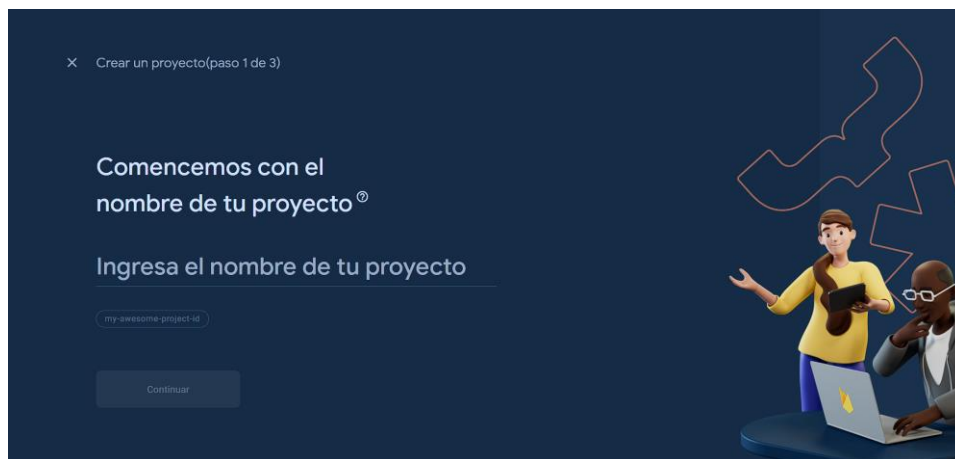


Ilustración 3. Asignación de nombre para el proyecto, Vilcacundo Jordy.

Una vez asignado el nombre daremos clic en continuar, luego de eso de forma opcional habilitaremos o deshabilitaremos Google analytics y daremos en continuar.

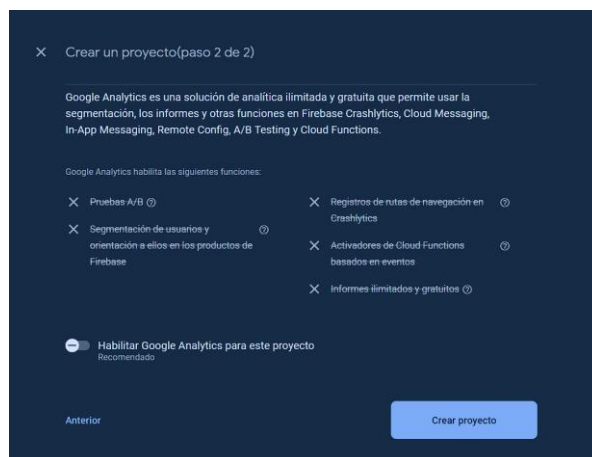


Ilustración 4. Selección de Google Analytics, Vilcacundo Jordy.

Una vez que deshabilitamos, daremos clic en crear proyecto.

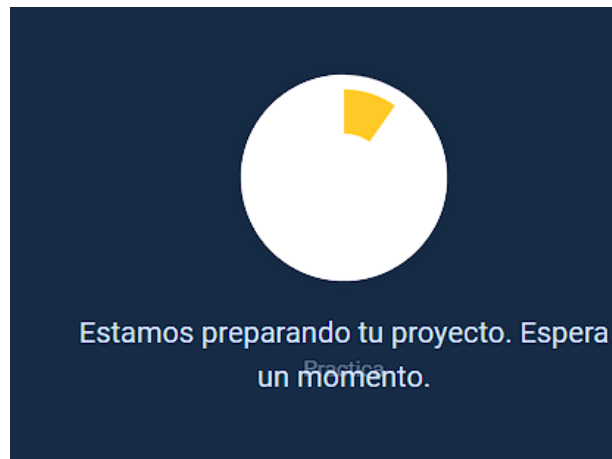


Ilustración 5. Pantalla de espera, Vilcacundo Jordy.

Una vez finalizado daremos clic en continuar.

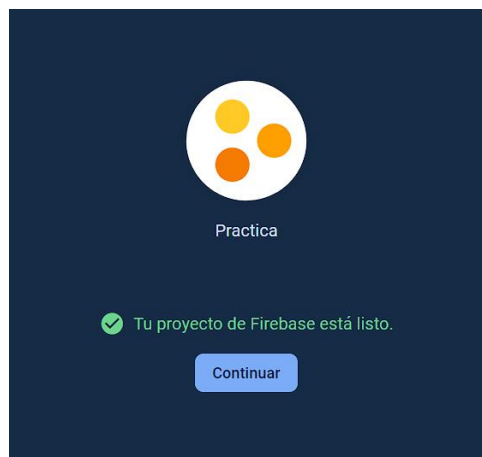


Ilustración 6. Confirmación de creación, Vilcacundo Jordy.

Una vez creado, nos redirigirá al repositorio, donde seleccionaremos el proyecto creado, una vez ingresamos nos mostrara la pantalla de inicio, daremos clic en el símbolo `</>`, el cual es para generar las KEY en web.

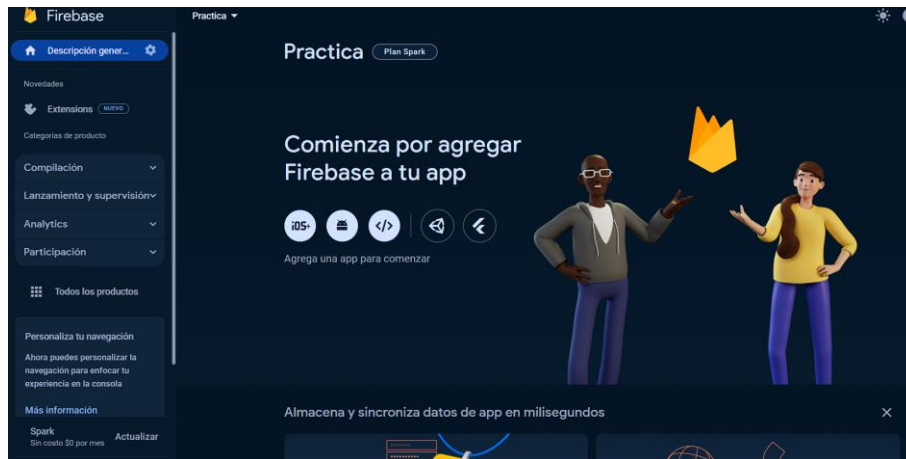


Ilustración 7. Pantalla de inicio de firebase, Vilcacundo Jordy.

2. Registro de la aplicación en firebase.

Para registrar una aplicación en firebase, daremos clic en `</>`, lo cual nos mostrara una ventana donde deberemos darle un nombre a nuestra aplicación, en nuestro caso es practica3

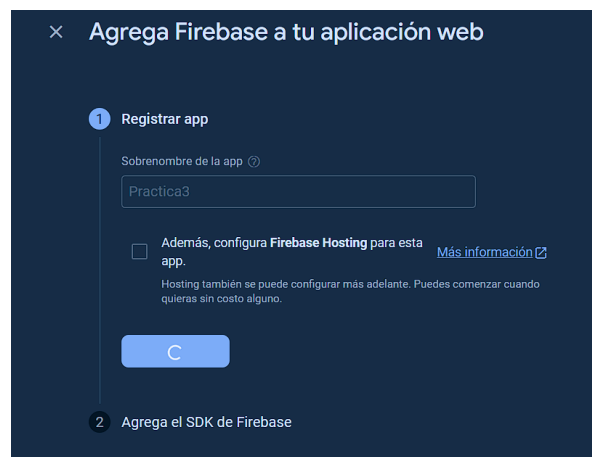


Ilustración 8. Registro de nombre de app, Vilcacundo Jordy.

Una vez que se registre la aplicación, nos generara el código que debemos utilizar para la conexión con FireBase.

Si ya usas [npm](#) o un agrupador de módulos como [Webpack](#) o [Rollup](#), puedes ejecutar el siguiente comando para instalar la versión más reciente del SDK (más información):

```
$ npm install firebase
```

Luego, inicializa Firebase y comienza a usar los SDK de los productos que quieres utilizar.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyBjpuKhjHamoh-fxm8frdjcHx8TpvCg8",
  authDomain: "practica-b426c.firebaseio.com",
  projectId: "practica-b426c",
  storageBucket: "practica-b426c.appspot.com",
  messagingSenderId: "715328637425",
  appId: "1:715328637425:web:f227fce9cc81b88988dc58"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Nota: Esta opción utiliza el SDK de JavaScript modular, que proporciona un tamaño reducido del SDK. Obtén más información sobre Firebase para la Web: [primeros pasos](#), [referencia de la API del SDK web](#) y [muestras](#).

[Ir a la consola](#)

Ilustración 9. SDK de firebase, Vilcacundo Jordy.

Una vez generado eso, daremos clic en ir a la consola y nos dirigiremos a Authentication.

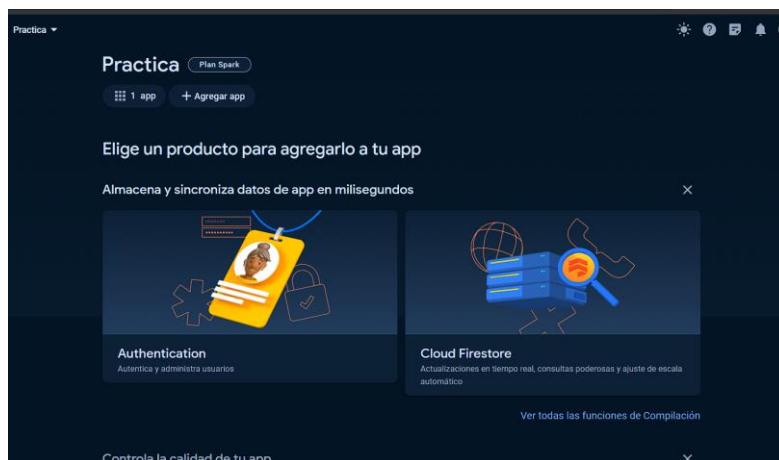


Ilustración 10. Consola de firebase, Vilcacundo Jordy.

3. Activación de métodos de autenticación

Una vez que ingresamos a Authentication, daremos clic en comenzar, donde se nos mostrara los métodos de inicio de sesión, seleccionaremos correo electrónico y contraseña, y Google.

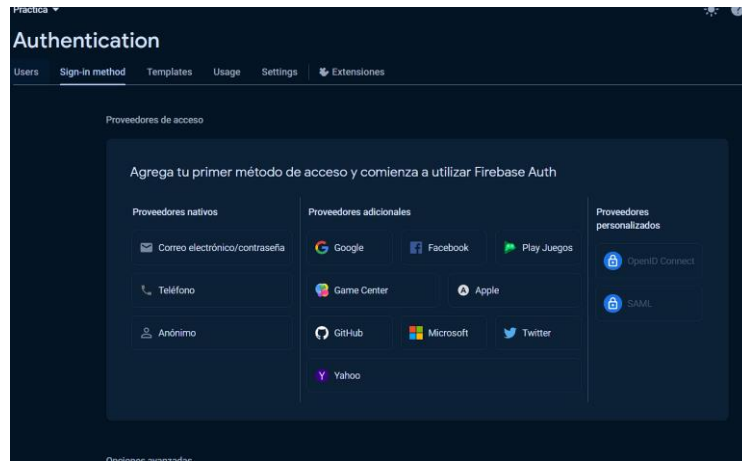


Ilustración 11. Métodos de inicio de sesión, Vilcacundo Jordy.

Una vez seleccionados y habilitados, regresamos a la consola, y nos dirigimos ahora a Cloud Firestore.

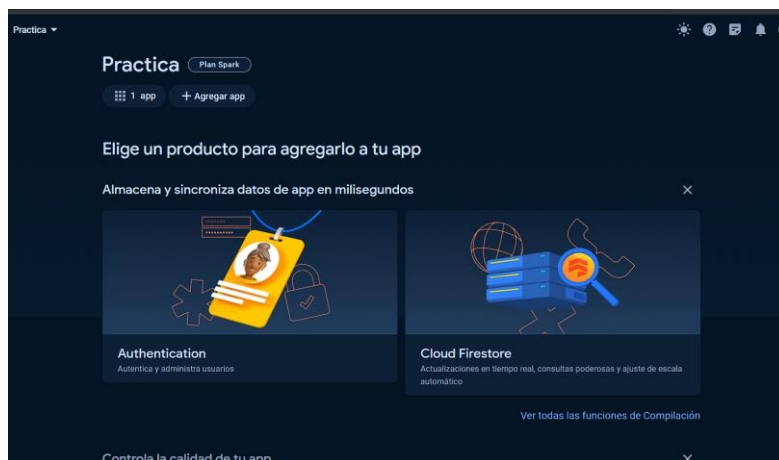


Ilustración 12. Consola de firebase, Vilcacundo Jordy.

4. Creación de una base de datos

Para crear una base de datos en firebase, debemos dirigirnos a cloud firestore, una vez dentro daremos clic en continuar, nos mostrar una ventana donde daremos siguiente y crearemos esa base de dato de forma temporal o en forma de prueba.

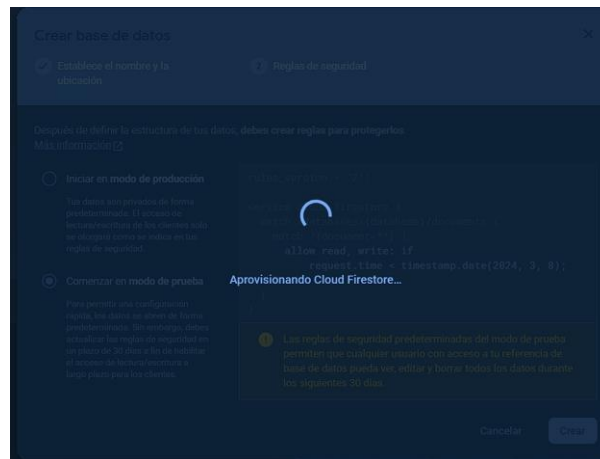


Ilustración 13. Creación de base de datos temporal, Vilcacundo Jordy.

Una vez creado, crearemos una colección, a la cual le colocaremos Users.

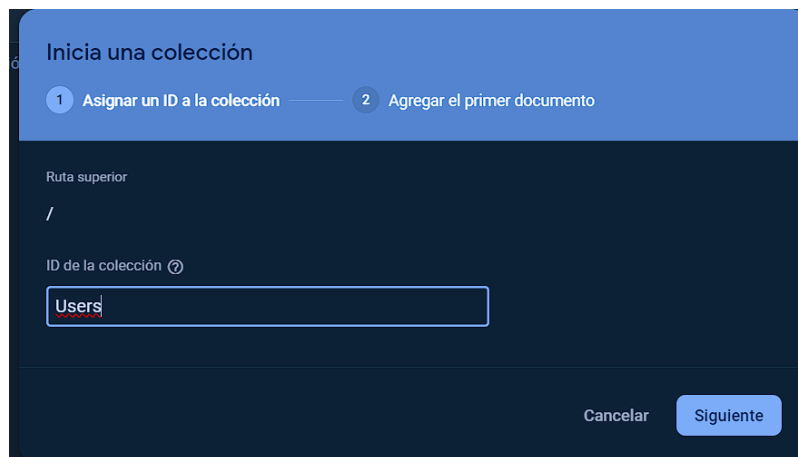


Ilustración 14. Creación de una nueva colección, Vilcacundo Jordy.

Luego agregaremos la primera tabla, la cual se llamará User, y tendrá los campos, Nombre, Apellido, Correo y Contraseña.

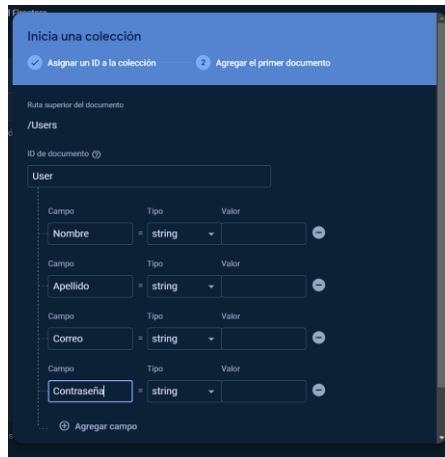


Ilustración 15. Creación de tabla y campos, Vilcacundo Jordy.

Una vez creado todo eso, se visualizara de la siguiente manera.

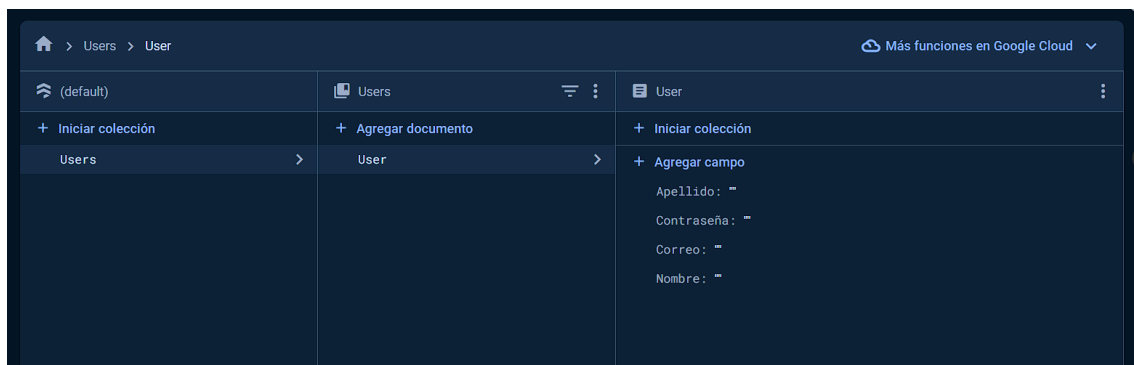


Ilustración 16. Base de datos Users, Vilcacundo Jordy.

5. Creación de archivo Firebase_config.js

En nuestro proyecto web, crearemos un archivo llamado Firebase_config.js el cual contendrá toda la configuración del firebase.

// Importa las funciones que necesitas de los SDKs que necesitas

```
import { initializeApp } from "firebase/app";
```

```
import { getAuth, GoogleAuthProvider } from "firebase/auth";
```

```
import { getFirestore } from "firebase/firestore";
```

// La configuración de tu aplicación web Firebase

```
const firebaseConfig = {
```

```
  apiKey: "AIzaSyBO7HFiWSisF0YjiYnEL59byJgD18w_uF0",
```

```

    authDomain: "practica3-d5f27.firebaseio.com",
    projectId: "practica3-d5f27",
    storageBucket: "practica3-d5f27.appspot.com",
    messagingSenderId: "1032778715648",
    appId: "1:1032778715648:web:e457d8b4d42de324e8bee5"
  };

  const app = initializeApp(firebaseConfig);
  const auth = getAuth(app);
  const googleProvider = new GoogleAuthProvider();
  const db = getFirestore(app);
  export { auth, googleProvider, db };

```

Este código es el que se nos ofrece cuando creamos el proyecto de firebase. Dentro le especificamos que queremos utilizar Google, firestore y auth.

6. Desarrollo de inicio de sesión Login.js

Una vez creado el archivo de configuración de firebase, procedemos a crear nuestro Login, donde importaremos las librerías de firebase que vamos a utilizar. Agregamos los campos de usuario y contraseña en caso de iniciar sesión con un usuario que se encuentre en la base de datos de firebase o podemos iniciar sesión mediante una cuenta de google. En caso de no tener ninguna de las dos tenemos un botón de registro.

```

import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { signInWithEmailAndPassword } from "firebase/auth";
import { signInWithPopup } from "firebase/auth";
import { auth, googleProvider } from './firebase-config';
import './Login.css';

```



```
const Login = () => {  
  
  const [email, setEmail] = useState("");  
  
  const [password, setPassword] = useState("");  
  
  const navigate = useNavigate();
```

```
  
  const signInWithGoogle = async () => {  
  
    try {  
  
      await signInWithPopup(auth, googleProvider);  
  
      navigate('/dashboard');  
  
    } catch (error) {  
  
      console.error(error.message);  
  
    }  
  
  };
```

```
  
  const loginUser = async (e) => {  
  
    e.preventDefault();  
  
    try {  
  
      await signInWithEmailAndPassword(auth, email, password);  
  
      navigate('/dashboard');  
  
    } catch (error) {  
  
      console.error(error.message);  
  
    }  
  
  };
```

```
return (  
  
  <div className="login-container">  
  
    <h2>Iniciar Sesión</h2>  
  
    <form onSubmit={loginUser}>  
  
      <div>  
  
        <input  
  
          type="email"  
  
          value={email}  
  
          onChange={(e) => setEmail(e.target.value)}  
  
          placeholder="Email"  
  
          required  
  
        />  
      </div>  
  
      <div>  
  
        <input  
  
          type="password"  
  
          value={password}  
  
          onChange={(e) => setPassword(e.target.value)}  
  
          placeholder="Contraseña"  
  
          required  
  
        />  
      </div>  
  
      <button type="submit">Iniciar Sesión</button>  
  
    </form>  
  </div>  
)
```

```

<button onClick={signInWithGoogle}>Iniciar sesión con Google</button>

      <p>¿No tienes una cuenta? <button onClick={() =>
navigate('/register')}}>Registrar</button></p>

</div>

);

};

export default Login;

```

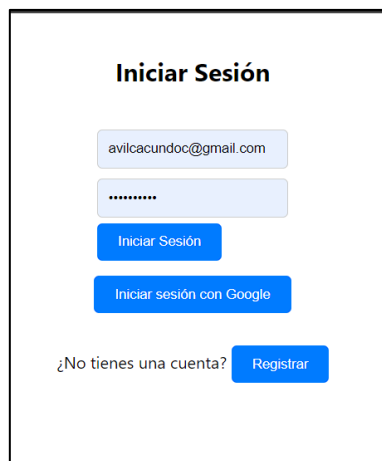


Ilustración 17. Interfaz de inicio de sesión, Vilcacundo Jordy.

7. Desarrollo de registro de usuario Register.js

En caso de que no cuente con un correo para iniciar mediante Google o no se encuentre en la base de datos, procedemos a registrar un nuevo usuario, para ello tenemos los campos de Nombre, Apellido, Correo y contraseña, una vez registrado iniciara sesión automáticamente.

```

import React, { useState } from 'react';

import { useNavigate } from 'react-router-dom';

import { auth, db } from './firebase-config';

import { createUserWithEmailAndPassword } from "firebase/auth";

import { collection, addDoc } from "firebase/firestore";

```

```
const Register = () => {

  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const navigate = useNavigate();

  const register = async (e) => {

    e.preventDefault();

    try {

      const userCredential = await createUserWithEmailAndPassword(auth, email,
password);

      console.log('Usuario registrado con éxito:', userCredential.user);

      await addDoc(collection(db, "User"), {

        uid: userCredential.user.uid,

        Nombre: firstName,

        Apellido: lastName,

        Correo: email,

      });

      console.log('Información del usuario guardada en Firestore.');
```

navigate('/dashboard');

```
    } catch (error) {

      console.error("Error al registrar el usuario:", error.message);
```

```

    alert(error.message);

  }

};

return (

  <div>

    <h2>Registro de Usuario</h2>

    <form onSubmit={register}>

      <input      type="text"      value={firstName}      onChange={(e)      =>
setFirstName(e.target.value)} placeholder="Nombre" required />

      <input      type="text"      value={lastName}      onChange={(e)      =>
setLastName(e.target.value)} placeholder="Apellido" required />

      <input type="email" value={email} onChange={(e) => setEmail(e.target.value)}
placeholder="Correo" required />

      <input      type="password"      value={password}      onChange={(e)      =>
setPassword(e.target.value)} placeholder="Contraseña" required />

      <button type="submit">Registrar</button>

    </form>

  </div>

);

};

export default Register;

```

Registro de Usuario

Nombre

Apellido

avilcacundoc@gmail.com

.....

Registrar

Ilustración 18. Interfaz de registro de usuarios, Vilcacundo Jordy.

Los usuarios registrados se podrán visualizar en la base de datos.

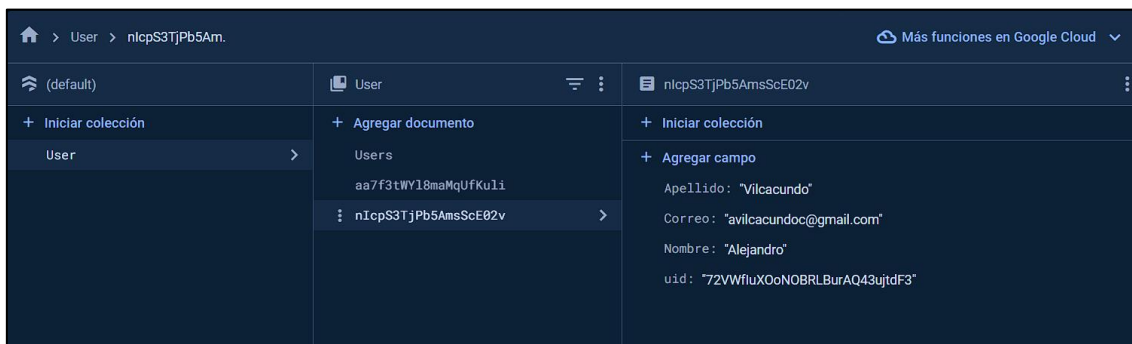


Ilustración 19. Usuarios registrados en la base de datos, Vilcacundo Jordy.

8. Vista de información Dashboard.js

Una vez que hayan iniciado sesión de forma correcta, se mostraran los datos como el correo electrónico o el nombre completo, junto con un botón para cerrar sesión.

```
import React from 'react';

import { useNavigate } from 'react-router-dom';

import { auth } from './firebase-config';

import './Dashboard.css';

const Dashboard = () => {

  const user = auth.currentUser;

  const navigate = useNavigate();

  const handleSignOut = () => {
```

```

    auth.signOut().then(() => {

        navigate('/');

    }).catch((error) => {

        console.error("Error al cerrar sesión:", error);

    });

};

return (

    <div className="dashboard-container">

        <h1 className="dashboard-welcome">Sesión iniciada correctamente</h1>

        <p className="dashboard-info">Bienvenido: {user ? user.email : ""}</p>

        <p className="dashboard-info">Nombre completo: {user ? user.displayName :
'Usuario sin nombre'}</p>

        <button    className="dashboard-button"    onClick={handleSignOut}>Cerrar
sesión</button>

    </div>

);

};

export default Dashboard;

```

INICIO DE SESION CON GOOGLE

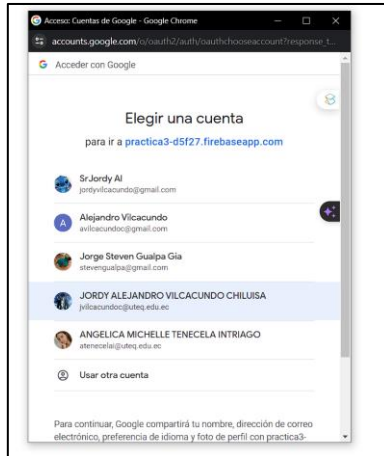


Ilustración 20: Inicio de Sesión con Google, Vilcacundo Jordy.

Cuando iniciamos sesión con google se nos mostrara que cuanta deseamos elegir para iniciar sesión, una vez que seleccionemos una nos mostrara el Dashboard con la información.

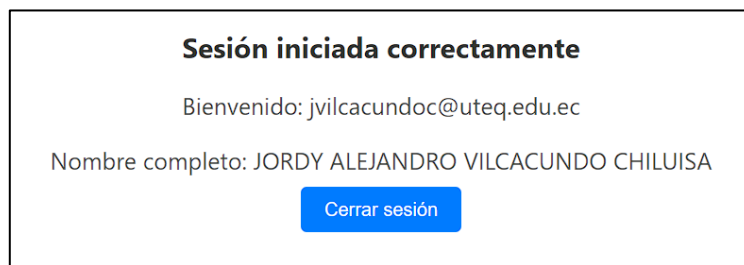


Ilustración 21. Información de usuario, Vilcacundo Jordy.

Inicio de sesión usando correo y contraseña

Cuando se inicia sesión mediante el uso de correo y contraseña, solo se mostrará el correo como información de usuario.

Iniciar Sesión

avilcacundoc@gmail.com

.....

Iniciar Sesión

Ilustración 22: Inicio de Sesión con credenciales, Vilcacundo Jordy.

Sesión iniciada correctamente

Bienvenido: avilcacundoc@gmail.com

Nombre completo:

Cerrar sesión

Ilustración 23: Sesión iniciada, Vilcacundo Jordy.

7.2 Migración de SQL SERVER a AZURE SQL

Problema: Se desea realizar la migración de la base de datos Northwind, desde SQL Server a un servidor en AZURE SQL, usando Azure Data Studio.

Descripción: Es importante realizar un análisis exhaustivo de la base de datos existente y de la aplicación que la utiliza. Esto incluye identificar las dependencias de la aplicación en la base de datos, las consultas y procedimientos almacenados utilizados, así como cualquier configuración específica de SQL Server que pueda necesitar ser ajustada para funcionar correctamente en Azure SQL.

Una vez completado el análisis, se puede proceder con la preparación de los datos para la migración. Esto puede implicar limpiar y optimizar la base de datos existente, así como también realizar cualquier conversión necesaria de tipos de datos o características específicas de SQL Server que no sean compatibles con Azure SQL.

Pasos:

1. Dentro de Sql Management Studio, seleccionar la base de datos a migrar.

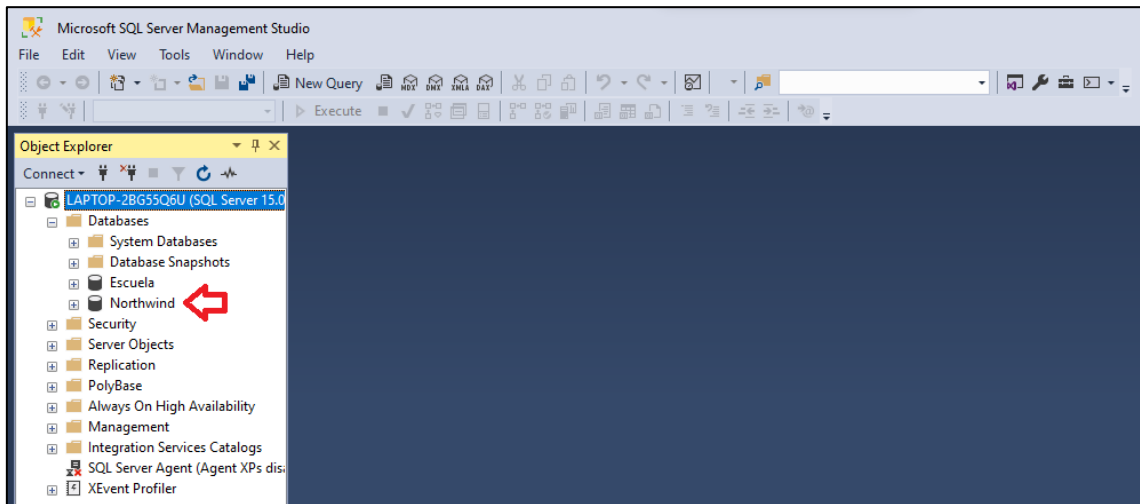


Ilustración 24: Selección de base de datos a migrar, Cedeño Orlando.

2. Seleccionar la opción de Export Data-tier Application (Exportación de aplicaciones de nivel de datos).

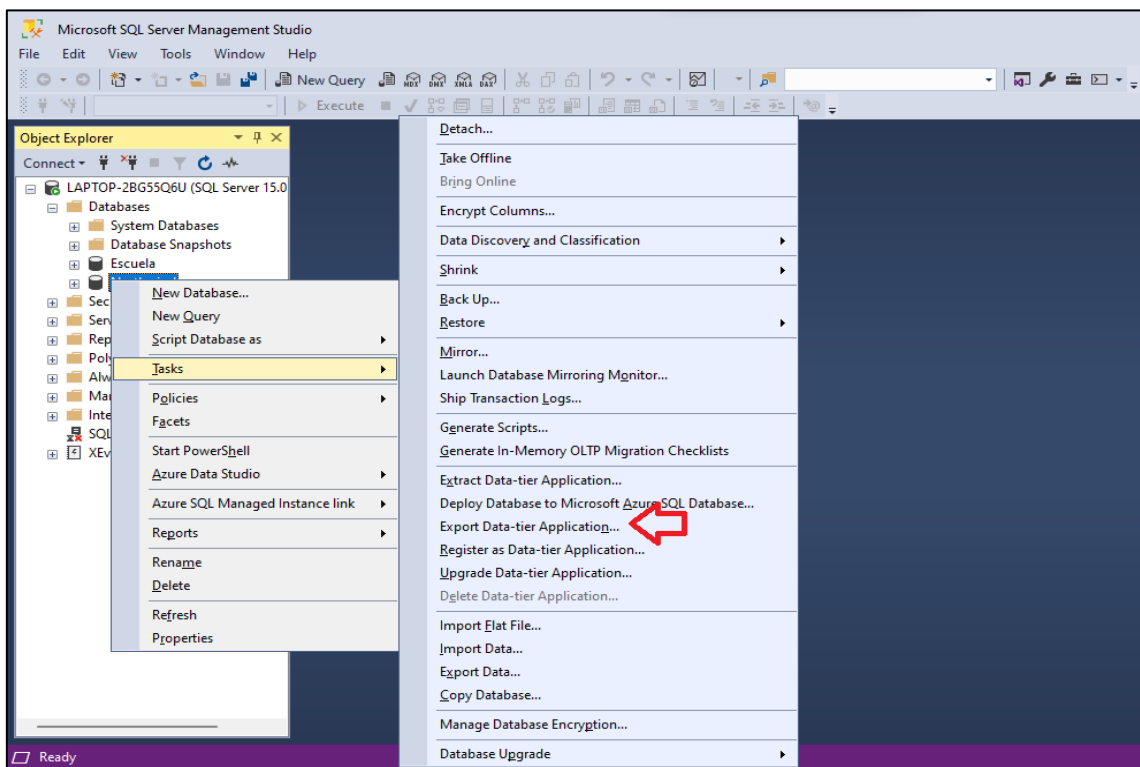


Ilustración 25: Export Data-tier Application, Cedeño Orlando.

3. Escoger la ubicación donde se guardará el archivo BACPAC de la base de datos.

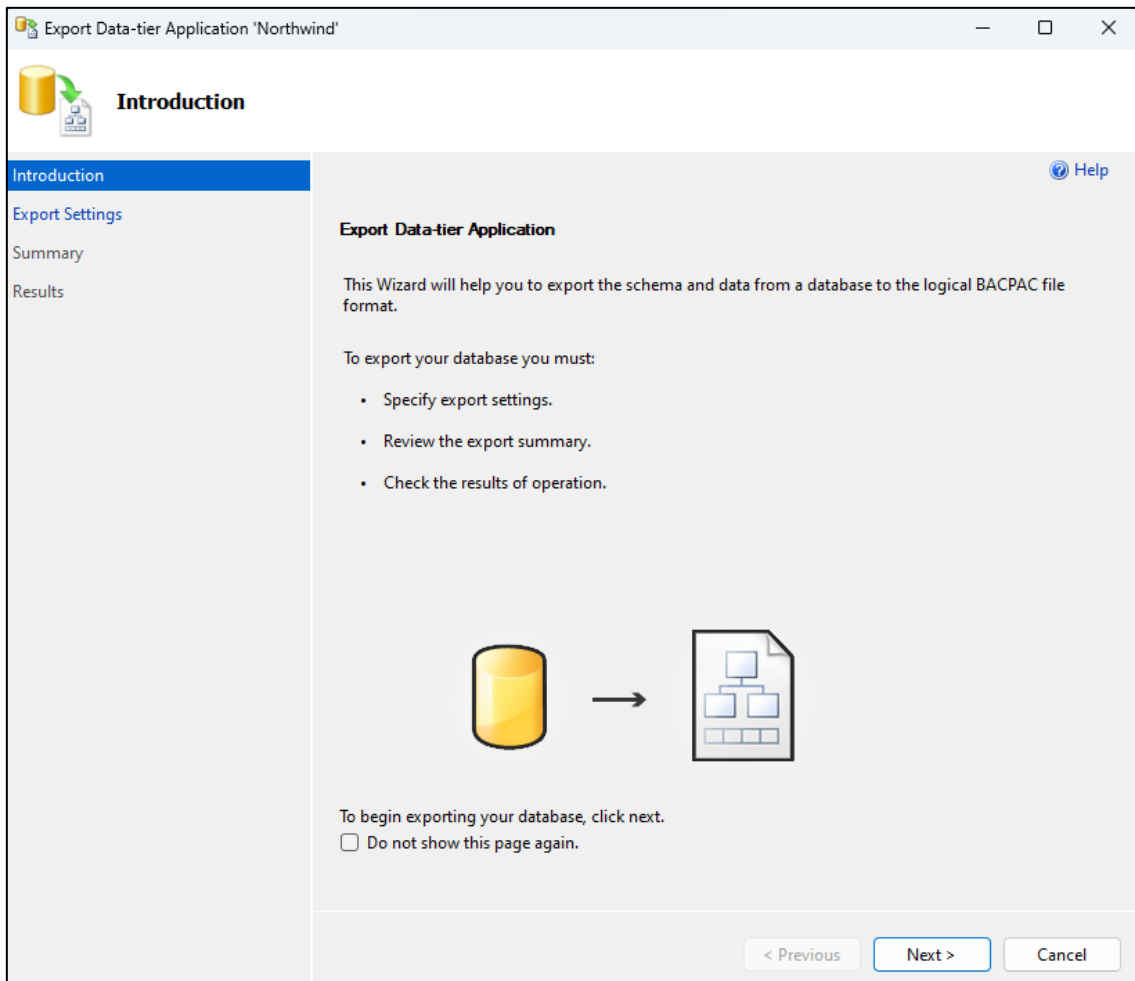


Ilustración 26: Creación de archivo Bacpac, Cedeño Orlando.

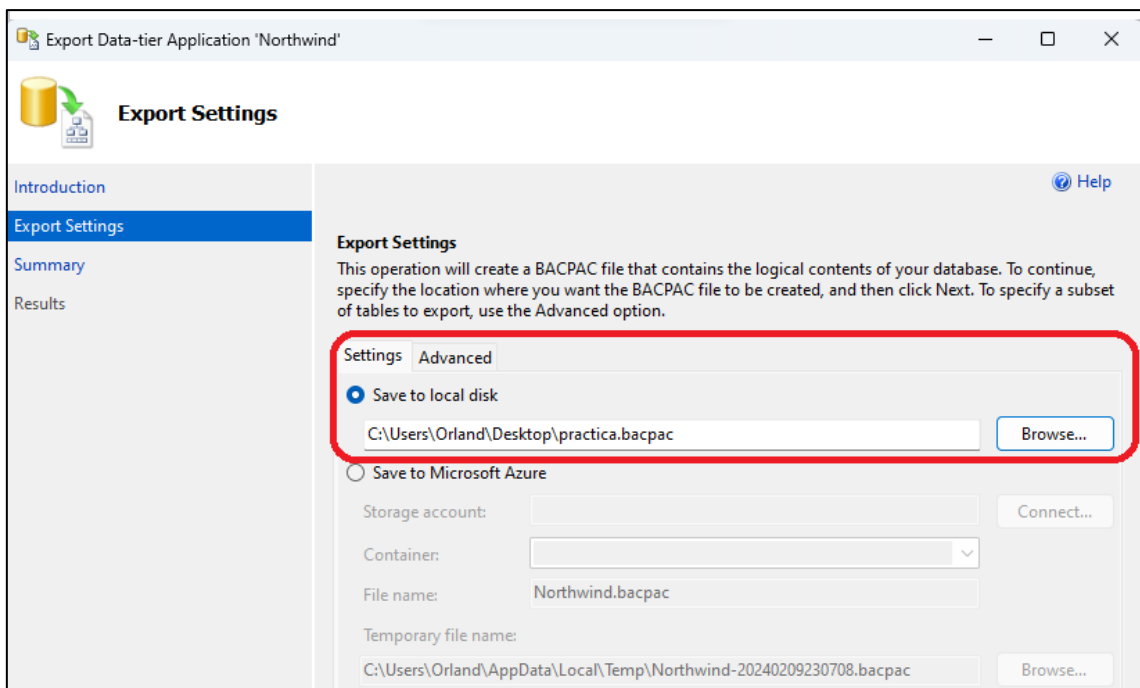
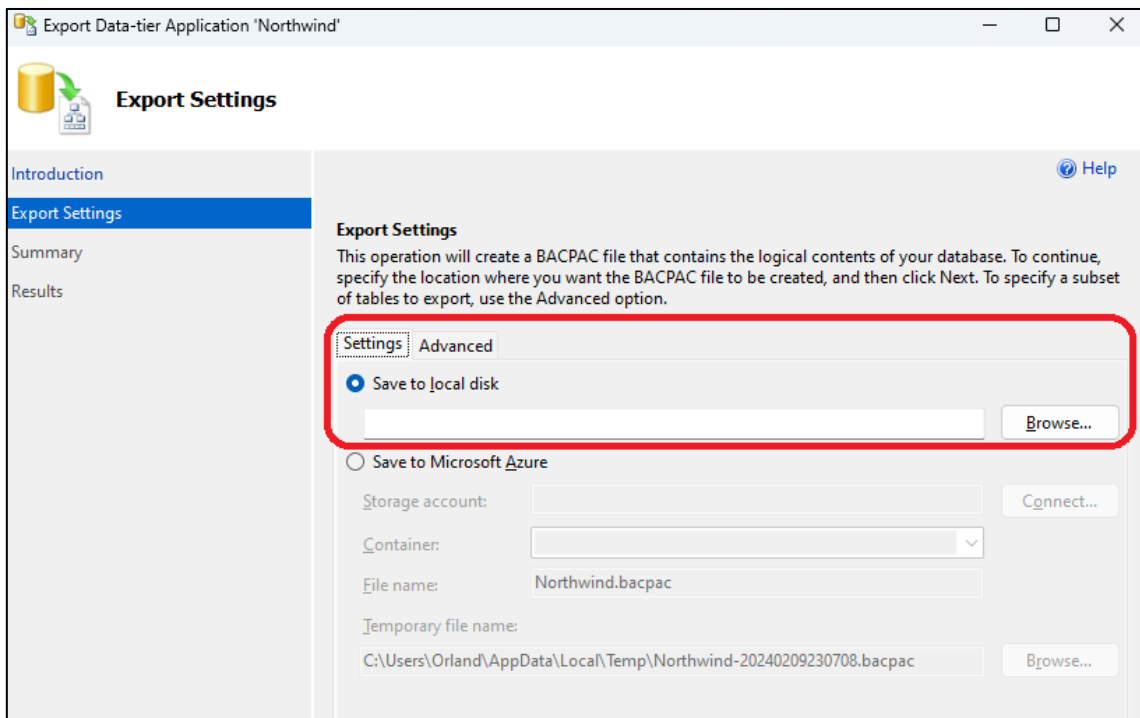


Ilustración 27: Exportar configuración de archivo Bacpac, Cedeño Orlando.

4. Finalizar la creación del archivo.

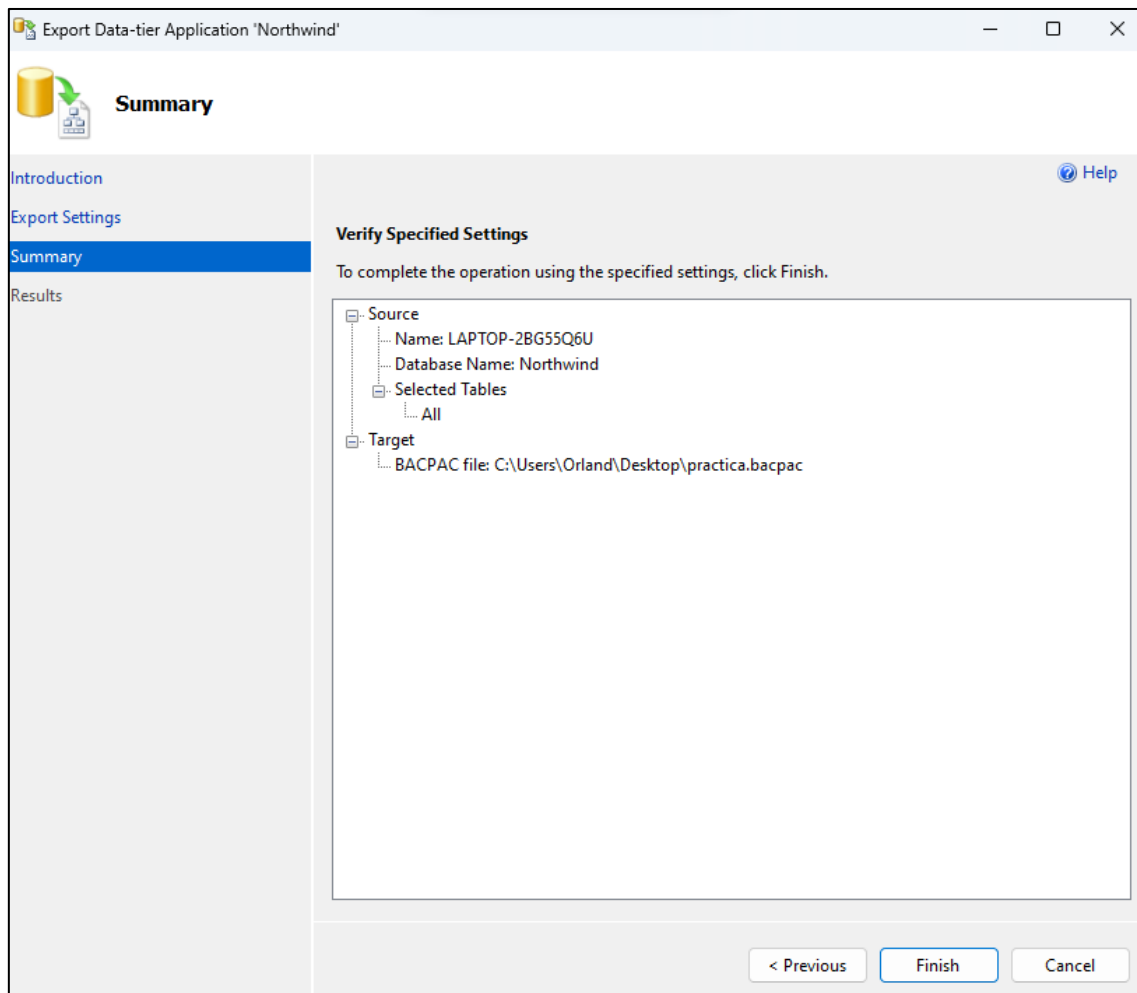


Ilustración 28: Archivo Bacpac creado, Cedeño Orlando.

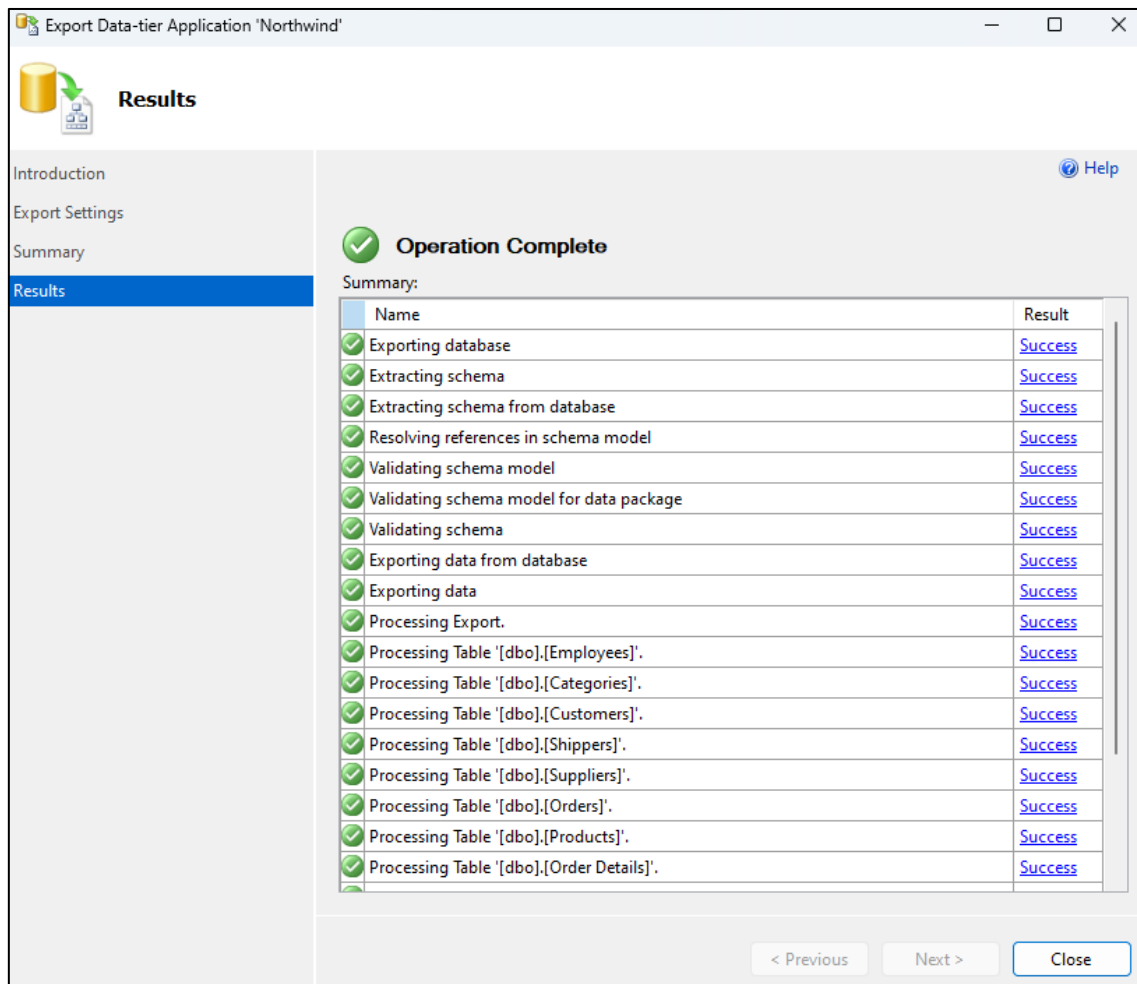


Ilustración 29: Comprobación de no existencia de errores al crear el archivo bacpac, Cedeño Orlando.

5. Dentro de Azure Data Studio, crear una nueva conexión con servidor localhost, y método de autenticación de Windows. Luego habilitar el certificado de servidor de confianza (enable trust server certificate).

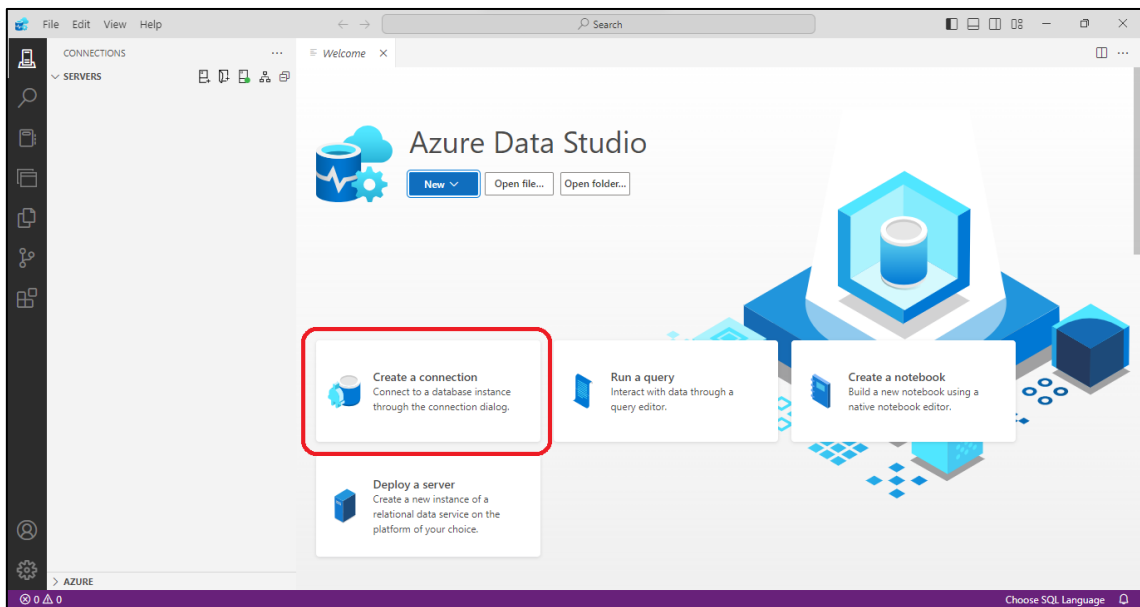


Ilustración 30: Creación de conexión en Azure Data Studio, Cedeño Orlando.

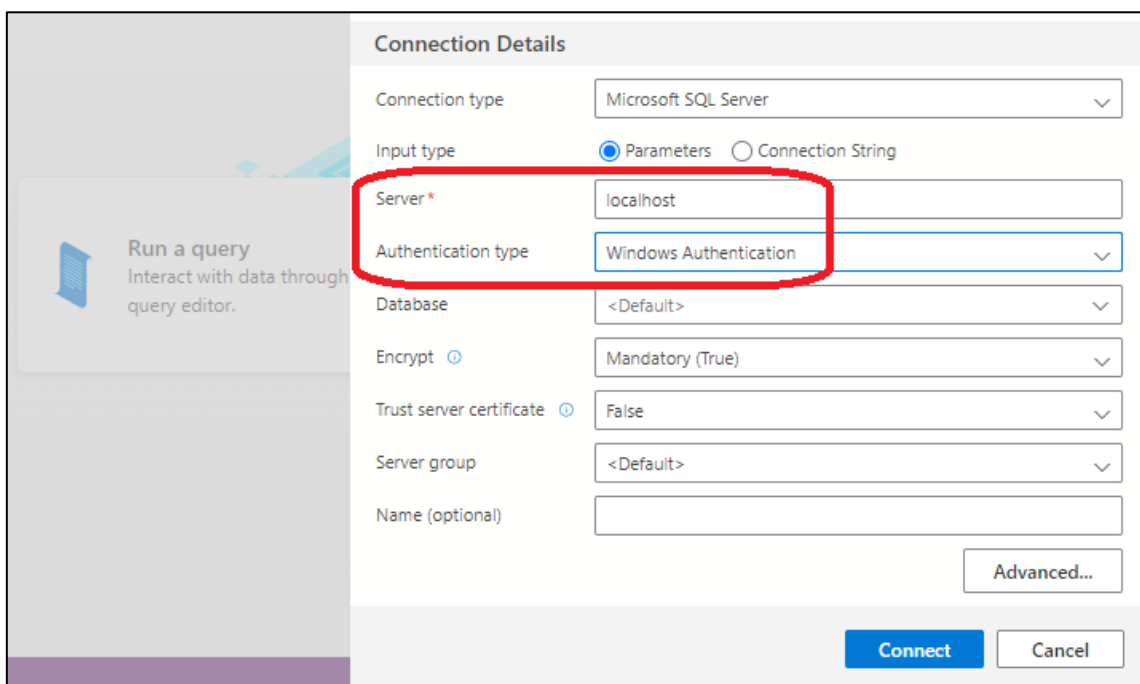


Ilustración 31: Configuración de Conexión, Cedeño Orlando.

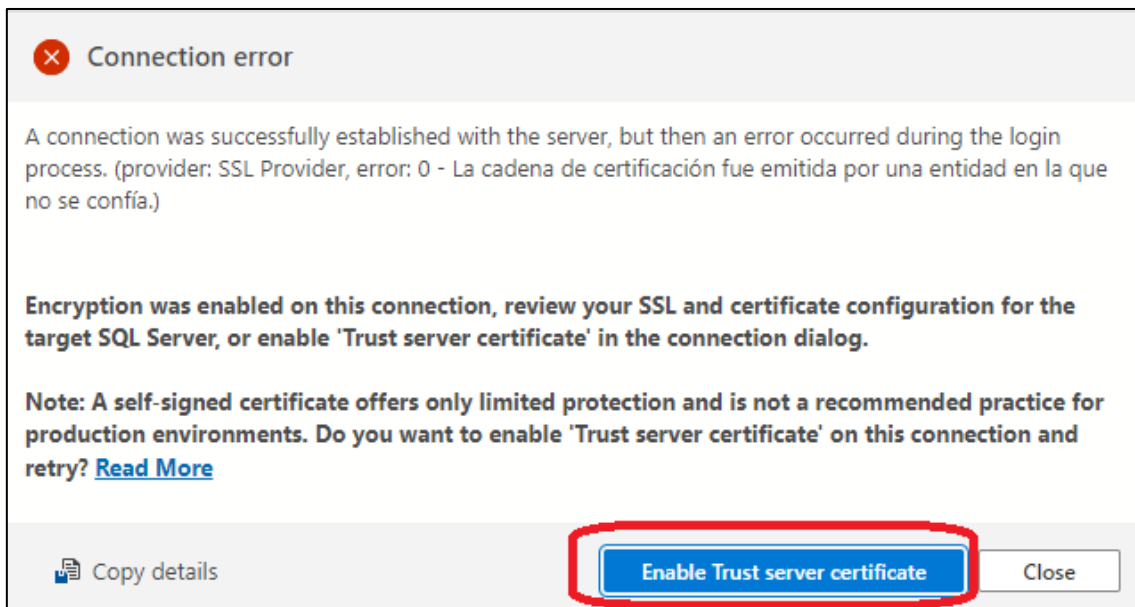


Ilustración 32: Aceptación de Certificado, Cedeño Orlando.

6. Instalar el complemento de SQL Server Dacpac, permitiendo la importación del archivo creado en SQL Server Management Studio.

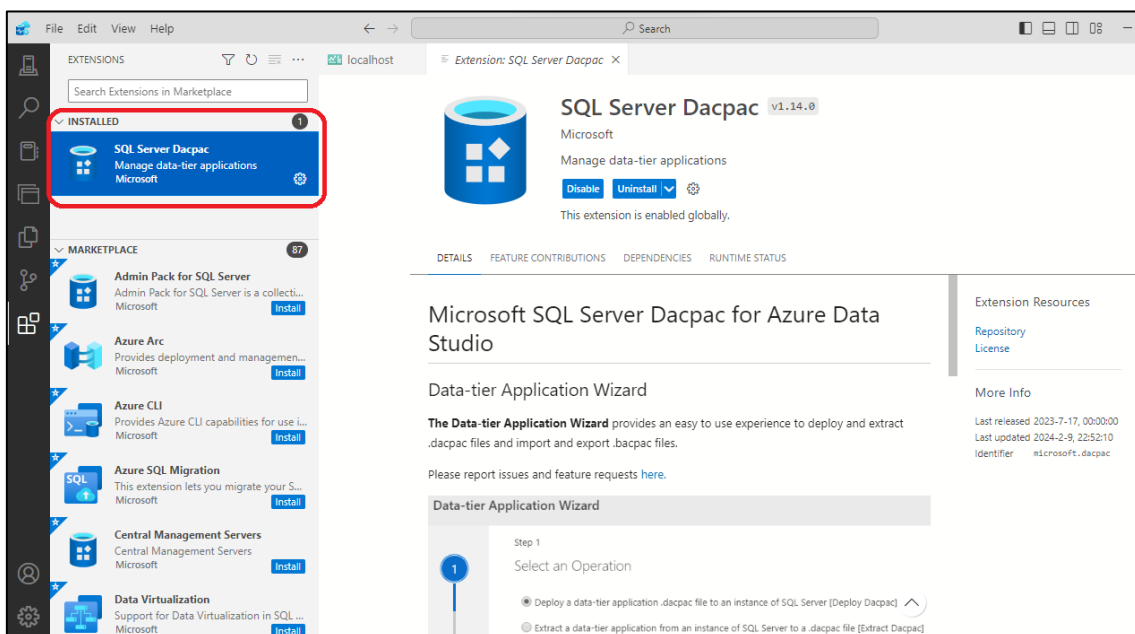


Ilustración 33: Instalación de complemento SQL Server Dacpac, Cedeño Orlando.

7. Dar click derecho en servidor y seleccionar la opción de Asistente para la aplicación de nivel de datos (Data-tier Application Wizard).

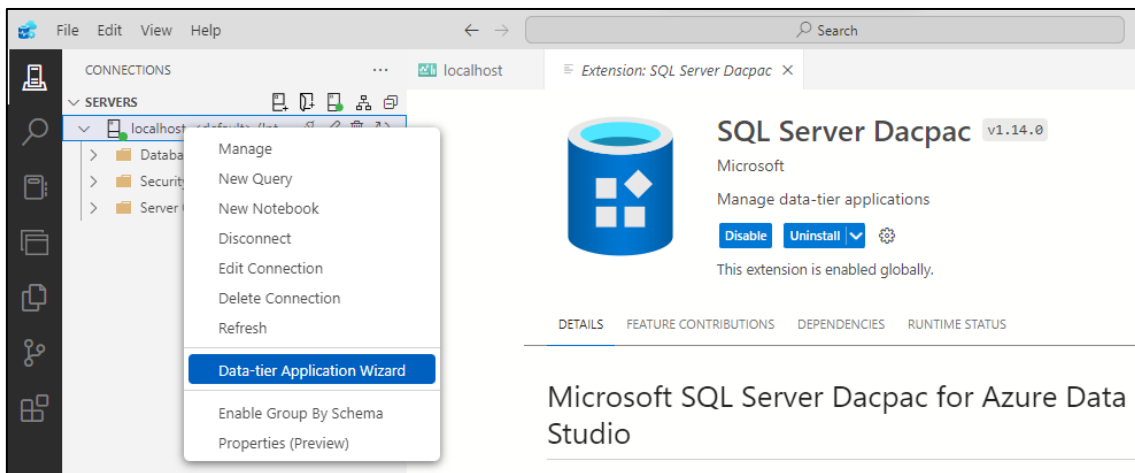


Ilustración 34: Uso de Data-tier Application Wizard, Cedeño Orlando.

8. Seleccionar Crear una base de datos a partir de un archivo .bacpac [Importar Bacpac] (Create a database from a .bacpac file [Import Bacpac]), y abrir el archivo que se creo en SQL Server Management Studio. Establecer el nombre del base de datos dentro de Azure Data Studio.

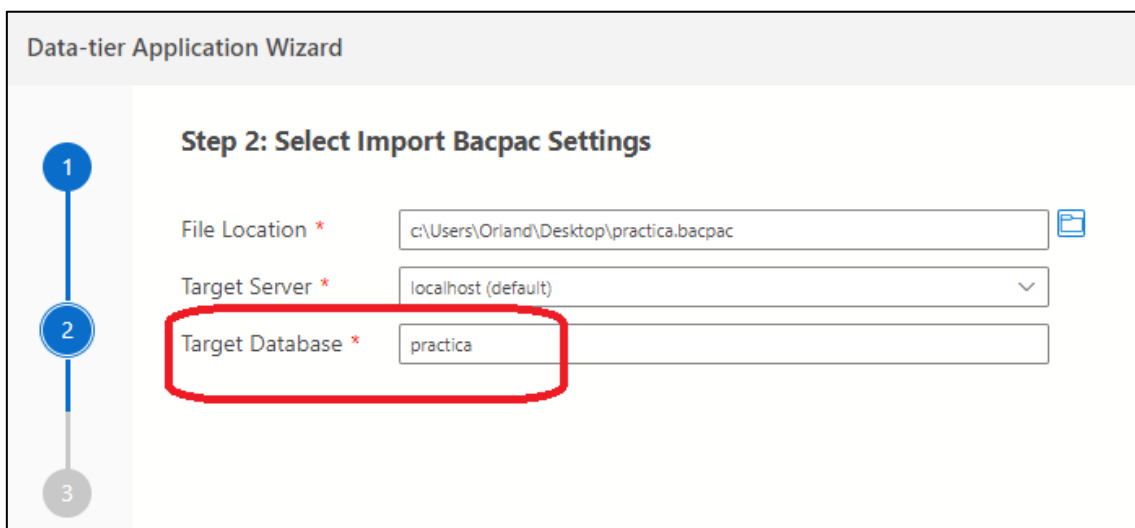


Ilustración 35: Abrir base de datos del archivo bacpac, Cedeño Orlando.

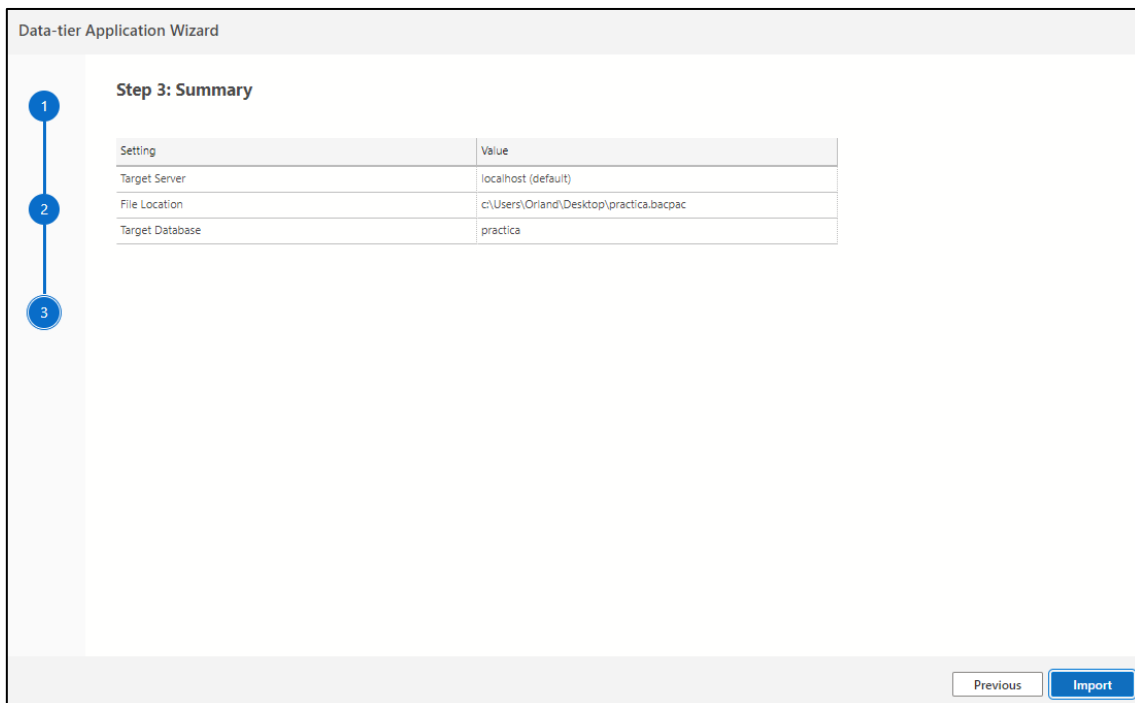


Ilustración 36: Comprobación de base de datos migrada, Cedeño Orlando.

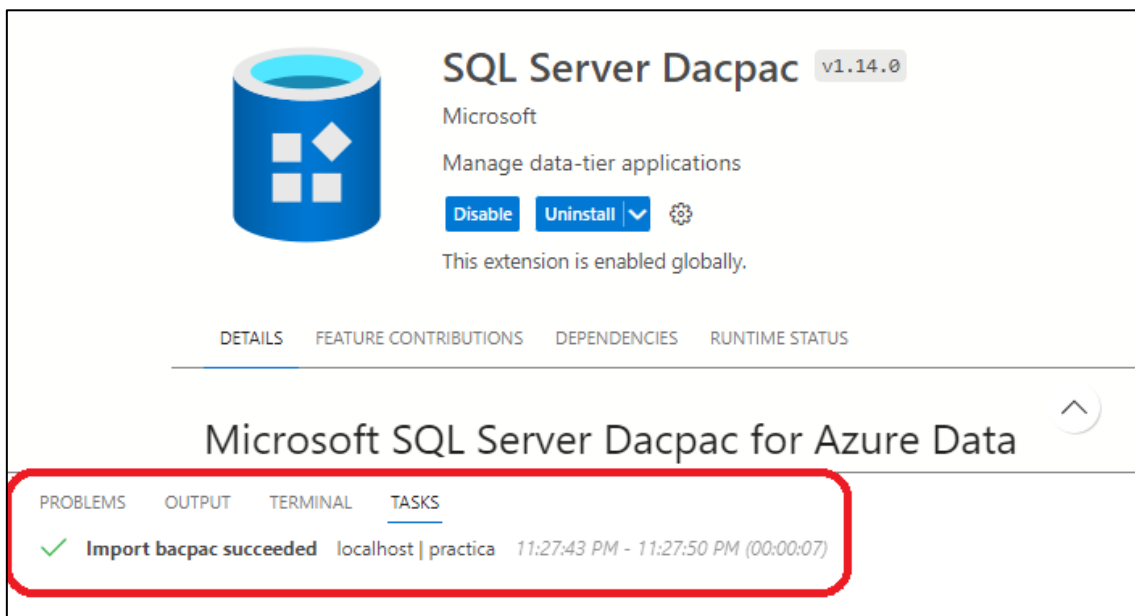


Ilustración 37: Base de datos migrada en Azure Sql, Cedeño Orlando.

9. Se comprueba la existencia de la base de datos migrada desde SQL Server Management Studio, realizar una consulta sobre los registros de la tabla Region.

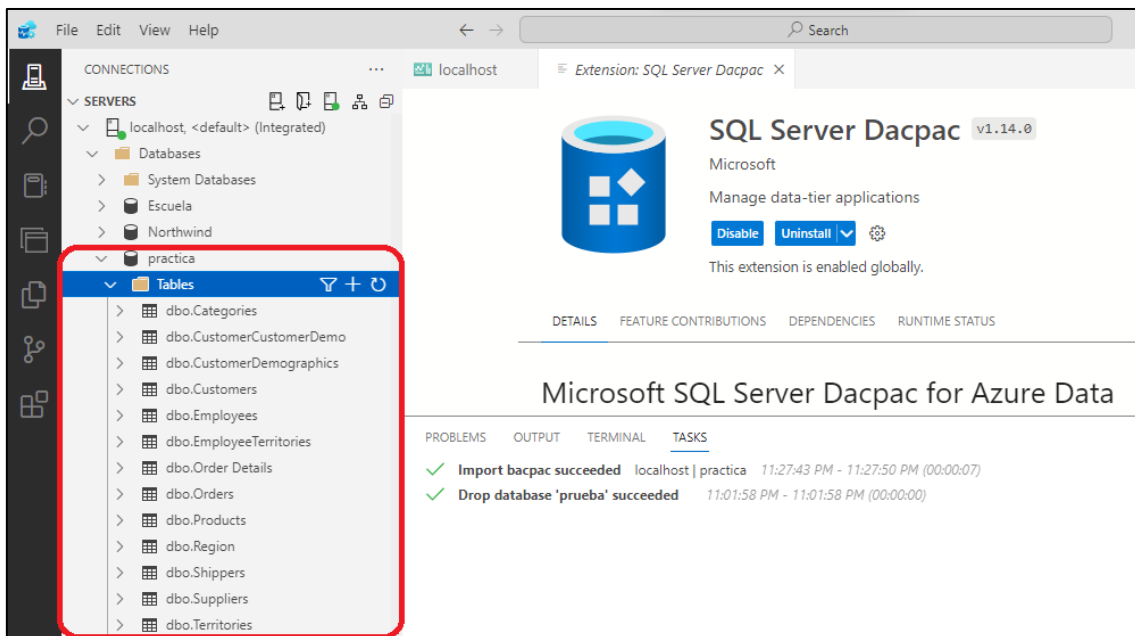


Ilustración 38: Tablas migradas, Cedeño Orlando.

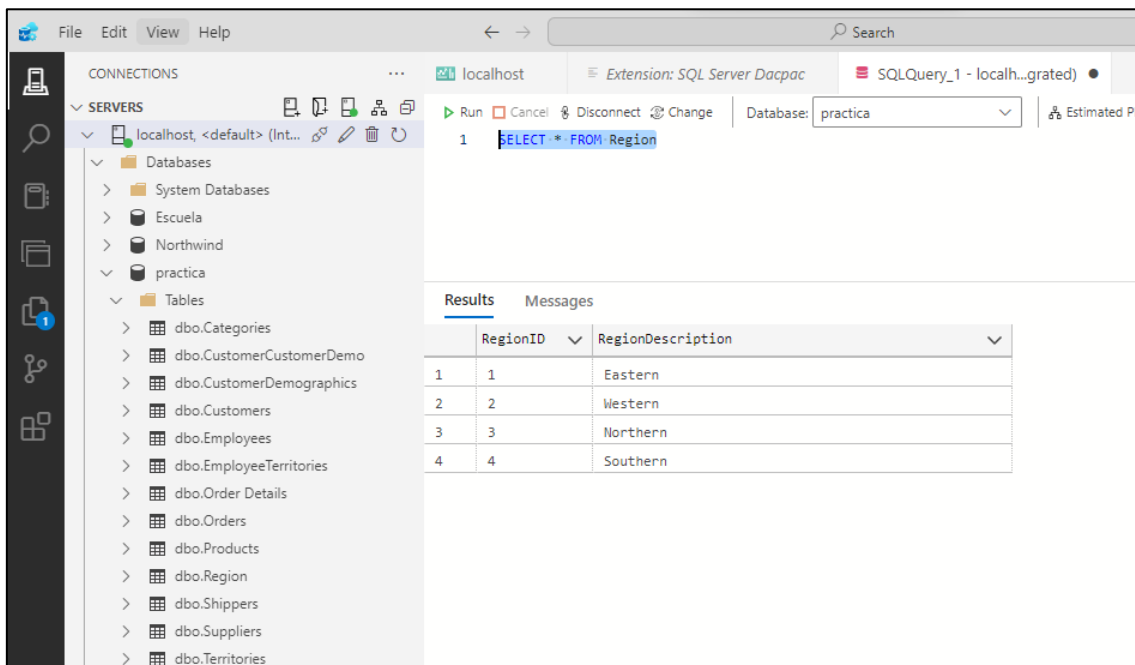


Ilustración 39: Consulta de registros de la tabla Region, Cedeño Salvatierra.

8. Conclusión

En conclusión, las plataformas en la nube como Firebase, Microsoft Azure y Amazon Web Services (AWS) ofrecen soluciones poderosas para el desarrollo de aplicaciones en capas. Firebase destaca por su integración completa y capacidades en tiempo real, ideal para aplicaciones que requieren actualizaciones instantáneas de datos y autenticación de usuarios segura. Mientras tanto, Microsoft Azure ofrece una amplia

gama de servicios en la nube, desde Azure App Service hasta Azure Functions, proporcionando flexibilidad y escalabilidad para proyectos web y móviles. AWS también se presenta como una opción sólida, con servicios como AWS Lambda y Amazon DynamoDB, ofreciendo un ecosistema completo para el desarrollo en la nube. La selección entre estas plataformas depende de las necesidades específicas del proyecto, pero todas ellas ofrecen herramientas que simplifican y agilizan el desarrollo de aplicaciones en capas en el entorno de la nube.

la migración de SQL Server a Azure SQL es un proceso complejo pero fundamental para muchas empresas que buscan modernizar su infraestructura tecnológica y aprovechar las ventajas de la computación en la nube. Al realizar esta tarea, es crucial seguir un enfoque planificado y meticuloso para garantizar el éxito y minimizar cualquier impacto negativo en la operación del negocio.

Durante la migración, es importante realizar un análisis exhaustivo de la base de datos existente y de la aplicación que la utiliza, así como también preparar los datos adecuadamente y planificar el proceso de migración con cuidado. La realización de pruebas exhaustivas después de la migración es esencial para garantizar que la aplicación funcione correctamente en Azure SQL y que los datos estén protegidos adecuadamente en la nube.

9. Link del Github

https://github.com/OrlandCede20/PRACT_DESARROCAPAS.git

10.Referencias

- [1] G. Banavar, D. Orr, and G. Lindstrom, "Layered, server-based support for object-oriented application development," in *Proceedings of International Workshop on Object Orientation in Operating Systems*, IEEE Comput. Soc. Press, pp. 2–11. doi: 10.1109/IWOOS.1995.470585.
- [2] G. Shchutska, "Development and application of the discrete model of multi-layered textile materials," *Eastern-European Journal of Enterprise Technologies*, vol. 6, no. 5 (84), pp. 39–45, Dec. 2016, doi: 10.15587/1729-4061.2016.85784.

- [3] Y. Kamimoto, N. Okamoto, T. Hagio, J. Yong-Jun, P. Deevanhxay, and R. Ichino, "Development of magnesium–iron layered double hydroxide and application to nitrate removal," *SN Appl Sci*, vol. 1, no. 11, p. 1399, Nov. 2019, doi: 10.1007/s42452-019-1240-7.
- [4] L. C. Sim, W. H. Yeo, J. Pubolaksono, L. H. Saw, J. Y. Tey, and C. H. Ting, "Development of analytical solution for thermo-mechanical stresses of multi-layered hollow cylinder for the application of underground hydrogen storage," *IOP Conf Ser Earth Environ Sci*, vol. 268, no. 1, p. 012018, Jun. 2019, doi: 10.1088/1755-1315/268/1/012018.
- [5] Y. Cheon, "Multiplatform Application Development for Android and Java," in *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*, IEEE, May 2019, pp. 1–5. doi: 10.1109/SERA.2019.8886800.
- [6] H. Esfahanizadeh, A. Cohen, M. Medard, and S. Shamai Shitz, "Distributed Computations with Layered Resolution," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, IEEE, Nov. 2022, pp. 257–261. doi: 10.1109/CloudNet55617.2022.9978858.
- [7] Y. S. Devi and L. Prabhakar, "Management of Possible Roles for Distributed Software Projects Using Layer Architecture," *International Journal of Information Technology and Computer Science*, vol. 7, no. 7, pp. 57–65, Jun. 2015, doi: 10.5815/ijitcs.2015.07.07.
- [8] Q. Sun, W. Ma, Y. Qiu, and W. Chen, "Layered Design and Collaborative Development of Network System and Its Application in Real-time Information Cluster," *International Journal of Computer Network and Information Security*, vol. 3, no. 4, pp. 25–31, Jun. 2011, doi: 10.5815/ijcnis.2011.04.04.
- [9] K. M. Konwar, N. Prakash, N. Lynch, and M. Médard, "A Layered Architecture for Erasure-Coded Consistent Distributed Storage," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, New York, NY, USA: ACM, Jul. 2017, pp. 63–72. doi: 10.1145/3087801.3087832.

- [10] I. Zyrianoff *et al.*, "Architecting and Deploying IoT Smart Applications: A Performance-Oriented Approach," *Sensors*, vol. 20, no. 1, p. 84, Dec. 2019, doi: 10.3390/s20010084.
- [11] Y. S. Devi and L. Prabhakar, "Management of Possible Roles for Distributed Software Projects Using Layer Architecture," *International Journal of Information Technology and Computer Science*, vol. 7, no. 7, pp. 57–65, Jun. 2015, doi: 10.5815/ijitcs.2015.07.07.
- [12] A. Bahtiar Semma, M. Ali, M. Saerozi, M. Mansur, and K. Kusriani, "Cloud computing: google firebase firestore optimization analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 29, no. 3, p. 1719, Mar. 2023, doi: 10.11591/ijeecs.v29.i3.pp1719-1728.
- [13] M. Sovian Salim Ibrahim, A. Faisol, and R. Primaswara Prasetya, "RANCANG BANGUN APLIKASI CONTROLING DAN MONITORING RUANGAN JEMURAN PINTAR MENGGUNAKAN FIREBASE PADA ARDUINO DAN ANDROID," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 7, no. 1, pp. 979–983, Jun. 2023, doi: 10.36040/jati.v7i1.6180.
- [14] G. Dhanush and D. K. Mohan, "INTEGRATED ADAPTABLE MOBILE BASED PLATFORM FOR UNIVERSITY/COLLEGE MANAGEMENT USING FIREBASE," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 1, pp. 373–378, May 2020, doi: 10.33564/IJEAST.2020.v05i01.062.
- [15] G. Dhanush and D. K. Mohan, "INTEGRATED ADAPTABLE MOBILE BASED PLATFORM FOR UNIVERSITY/COLLEGE MANAGEMENT USING FIREBASE," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 1, pp. 373–378, May 2020, doi: 10.33564/IJEAST.2020.v05i01.062.
- [16] R. Y. Wijaya, N. A. Saputra, and I. B. Trisno, "Travel Journal Application as an Android-Based Traveling Visitor Using Firebase," *JISA (Jurnal Informatika dan Sains)*, vol. 6, no. 1, pp. 8–13, Jun. 2023, doi: 10.31326/jisa.v6i1.1446.
- [17] A. A. Shonta, L. N. Hamidah, M. Hasan, M. M. Dewi, Y. Astuti, and I. R. Wulandari, "Penerapan Firebase Realtime Database Pada Aplikasi Media Informasi dan

Pendaftaran Training IT Berbasis Android,” *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 6, no. 3, p. 1517, Jul. 2022, doi: 10.30865/mib.v6i3.4040.

- [18] C. Khawas and P. Shah, “Application of Firebase in Android App Development- A Study,” *Int J Comput Appl*, vol. 179, no. 46, pp. 49–53, Jun. 2018, doi: 10.5120/ijca2018917200.
- [19] M. M. Engel, “Android Based Thesis Mentoring System Using Google Firebase,” *ComTech: Computer, Mathematics and Engineering Applications*, vol. 9, no. 2, p. 73, Dec. 2018, doi: 10.21512/comtech.v9i2.4992.
- [20] F. Afira and J. W. Simatupang, “Real-Time Web-based Dashboard using Firebase for Automated Object Detection Applied on Conveyor,” *Green Intelligent Systems and Applications*, vol. 3, no. 1, pp. 35–47, Jun. 2023, doi: 10.53623/gisa.v3i1.251.
- [21] D. Sharma and H. Dand, “Firebase as BaaS for College Android Application,” 2019.
- [22] M. M. Engel, “Android Based Thesis Mentoring System Using Google Firebase,” *ComTech: Computer, Mathematics and Engineering Applications*, vol. 9, no. 2, p. 73, Dec. 2018, doi: 10.21512/comtech.v9i2.4992.
- [23] V. Harianto, P. Tjandra, and N. Setiyawati, “PERANCANGAN APLIKASI E-VOTING BERBASIS ANDROID DENGAN TEKNOLOGI FIREBASE (STUDI KASUS: PEMILIHAN KETUA HMP FTI UKSW).” [Online]. Available: <http://www.jurnal.umk.ac.id/sitech>
- [24] “INTEGRATED ADAPTABLE MOBILE BASED PLATFORM FOR UNIVERSITY/COLLEGE MANAGEMENT USING FIREBASE”.
- [25] R. Riaz, N. U. A. Gillani, S. Rizvi, S. Shokat, and S. J. Kwon, “SUBBASE: An authentication scheme for wireless sensor networks based on user biometrics,” *Wirel Commun Mob Comput*, vol. 2019, 2019, doi: 10.1155/2019/6370742.
- [26] A. Suryaningrat, D. Kurnianto, and F. T. Syifa, “PEMANFAATAN GOOGLE FIREBASE PADA SISTEM TEMPAT SAMPAH PINTAR BERBASIS INTERNET OF THINGS UTILIZATION OF GOOGLE FIREBASE ON SMART TRASH BIN SYSTEM

BASED ON INTERNET OF THINGS.” [Online]. Available:
<http://dinarek.unsoed.ac.id>

- [27] A. R. Wiratno and K. Hastuti, “Implementation of Firebase Realtime Database to track BRT Trans Semarang,” *Scientific Journal of Informatics*, vol. 4, no. 2, pp. 2407–7658, 2017, [Online]. Available: <http://journal.unnes.ac.id/nju/index.php/sji>
- [28] F. Afira and J. W. Simatupang, “Real-Time Web-based Dashboard using Firebase for Automated Object Detection Applied on Conveyor,” *Green Intelligent Systems and Applications*, vol. 3, no. 1, pp. 35–47, Jun. 2023, doi: 10.53623/gisa.v3i1.251.
- [29] K. A. Nugraha, “JEPIN (Jurnal Edukasi dan Penelitian Informatika) Basis Data Awan Non-Relasional Firestore untuk Penyimpanan Data Pesan”.
- [30] T. C. Phan, N. D. K. Quach, T. T. Nguyen, T. T. Nguyen, J. Jo, and Q. V. H. Nguyen, “Real-time wildfire detection with semantic explanations,” *Expert Syst Appl*, vol. 201, Sep. 2022, doi: 10.1016/j.eswa.2022.117007.
- [31] R. Yohanes Wijaya, N. A. Saputra, and I. B. Trisno, “JISA (Jurnal Informatika dan Sains) Travel Journal Application as an Android-Based Traveling Visitor Using Firebase,” 2023.
- [32] R. Barber *et al.*, “Evolving Databases for New-Gen Big Data Applications.”
- [33] H. Zheng, J. Duan, Y. Dong, and Y. Liu, “Real-time fire detection algorithms running on small embedded devices based on MobileNetV3 and YOLOv4,” *Fire Ecology*, vol. 19, no. 1, Dec. 2023, doi: 10.1186/s42408-023-00189-0.
- [34] T. F. Prasetyo, R. Rohmat, and D. Zalilluddin, “Design and Build Disaster Emergency Response Systems Using Firebase Cloud Messaging Based on Android and SMS Gateway,” *Jurnal Informatika*, vol. 13, no. 1, p. 16, Jan. 2019, doi: 10.26555/jifo.v13i1.a11664.
- [35] L. de O. Turci, “Real-time operating system FreeRTOS application for fire alarm project in reduced scale,” *International Journal of Computing and Digital Systems*, vol. 6, no. 4, pp. 197–204, Jul. 2017, doi: 10.12785/IJCDS/060405.

- [36] Y. Li, J. Shang, M. Yan, B. Ding, and J. Zhong, "Real-Time Early Indoor Fire Detection and Localization on Embedded Platforms with Fully Convolutional One-Stage Object Detection," *Sustainability (Switzerland)*, vol. 15, no. 3, Feb. 2023, doi: 10.3390/su15031794.
- [37] D. Boedi Prasetyo, H. Himawan, W. Kaswidjanti, and F. Zeaneth Universitas Pembangunan Nasional Veteran Yogyakarta, "Prototype Design of IoT Remote Monitoring System for Industrial Process Using Firebase Realtime Database," *Yogyakarta Conference Series Proceeding on Engineering and Science Series (ESS)*, vol. 1, no. 1, pp. 129–138, 2020, doi: 10.31098/ess.v1i1.103.
- [38] P. Gupta, "Part of the Information Security Commons Citation Citation GUPTA, Payas. Exploiting Human Factors in User Authentication," 2013. [Online]. Available: https://ink.library.smu.edu.sg/etd_coll
- [39] B. L. Ortiz, V. Gupta, J. W. Chong, K. Jung, and T. Dallas, "User Authentication Recognition Process Using Long Short-Term Memory Model," *Multimodal Technologies and Interaction*, vol. 6, no. 12, Dec. 2022, doi: 10.3390/mti6120107.
- [40] H. Chen, L. Ge, and L. Xie, "A user authentication scheme based on elliptic curves cryptography for wireless ad hoc networks," *Sensors*, vol. 15, no. 7, pp. 17057–17075, Jul. 2015, doi: 10.3390/s150717057.
- [41] N. Nguyen and S. Sigg, "User Authentication based on Personal Image Experiences," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2018*, Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 336–341. doi: 10.1109/PERCOMW.2018.8480087.
- [42] D. Jayadi and U. Darusalam, "Pengembangan Sistem Informasi Peminjaman Alat Laboratorium Berbasis Android dan Realtime Database Menerapkan Framework FAST," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 6, no. 1, p. 424, Jan. 2022, doi: 10.30865/mib.v6i1.3495.
- [43] A. R. Wiratno and K. Hastuti, "Implementation of Firebase Realtime Database to track BRT Trans Semarang," *Scientific Journal of Informatics*, vol. 4, no. 2, pp.

2407-7658, 2017, [Online]. Available:
<http://journal.unnes.ac.id/nju/index.php/sji>

- [44] J. A. Pratama, R. M. Negara, N. Bogi, and A. Karna, "WEBSITE AND DATABASE IMPLEMENTATION FOR VEHICLE VIOLATION DATA CHECK BASED ON GOOGLE FIREBASE."
- [45] M. Microcontroller *et al.*, "PERANCANGAN PROTOTYPE ALAT PENANGKAP IKAN PADA TAMBAK THE DESIGN OF A FISH CATCHER PROTOTYPE IN A FISH FARM USING MICROCONTROLLER AND A REALTIME DATABASE."
- [46] X. Li, Z. Li, X. Ma, and C. Liu, "A novel HBase data storage in wireless sensor networks," *EURASIP J Wirel Commun Netw*, vol. 2017, no. 1, Dec. 2017, doi: 10.1186/s13638-017-0827-1.
- [47] F. Suryatini and F. Ilman Fauzandi, "SISTEM AKUISISI DATA SUHU DAN KELEMBABAN TANAH PADA IRIGASI TETES OTOMATIS BERBASIS INTERNET OF THINGS," 2018.
- [48] R. Ahmad Ma and N. Hayati, "Sistem Monitoring Tempat Sampah Pintar Secara Real-time Menggunakan Metode Fuzzy Logic Berbasis IOT," *Jurnal Infomedia*, vol. 4, no. 2, 2019.
- [49] A. B. Semma, M. Ali, M. Saerozi, Mansur, and Kusrini, "Cloud computing: google firebase firestore optimization analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 29, no. 3, pp. 1719-1728, Mar. 2023, doi: 10.11591/ijeecs.v29.i3.pp1719-1728.
- [50] R. T. Y. R. B. Roosevelt Joshua Gunadi, "PENERAPAN FIREBASE CLOUD STORAGE PADA APLIKASI MOBILE ANDROID UNTUK MELAKUKAN PENYIMPANAN IMAGE LAHAN PERTANIAN".
- [51] J. Soh, M. Copeland, A. Puca, and M. Harris, *Microsoft Azure*. Berkeley, CA: Apress, 2020. doi: 10.1007/978-1-4842-5958-0.
- [52] P. Mazumdar, S. Agarwal, and A. Banerjee, "Introduction to Microsoft Azure," in *Pro SQL Server on Microsoft Azure*, Berkeley, CA: Apress, 2016, pp. 1-17. doi: 10.1007/978-1-4842-2083-2_1.

- [53] Y. Zhu *et al.*, "Towards Building Autonomous Data Services on Azure," in *Companion of the 2023 International Conference on Management of Data*, New York, NY, USA: ACM, Jun. 2023, pp. 217–224. doi: 10.1145/3555041.3589674.
- [54] S. K, A. X. K, D. Davis, and N. Jayapandian, "Internet of Things and Cloud Computing Involvement Microsoft Azure Platform," in *2022 International Conference on Edge Computing and Applications (ICECAA)*, IEEE, Oct. 2022, pp. 603–609. doi: 10.1109/ICECAA55415.2022.9936126.
- [55] M. E. Rana and V. Mothi, "Cloud Computing as an Enabler in the Mobile Application Domain," in *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, IEEE, Oct. 2022, pp. 184–189. doi: 10.1109/ICDABI56818.2022.10041462.
- [56] I. Baldini *et al.*, "The serverless trilemma: function composition for serverless computing," in *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, New York, NY, USA: ACM, Oct. 2017, pp. 89–103. doi: 10.1145/3133850.3133855.
- [57] A. Hall and U. Ramachandran, "An execution model for serverless functions at the edge," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, New York, NY, USA: ACM, Apr. 2019, pp. 225–236. doi: 10.1145/3302505.3310084.
- [58] J. G. Quenum and J. Josua, "Multi-cloud serverless function composition," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing*, New York, NY, USA: ACM, Dec. 2021, pp. 1–10. doi: 10.1145/3468737.3494090.
- [59] S. Kumar.N and S. Samy.S, "Serverless Computing To Predict Cold Start In Azure And Monitoring I/O Read And Write Using Machine Learning," in *2023 9th International Conference on Smart Structures and Systems (ICSSS)*, IEEE, Nov. 2023, pp. 1–6. doi: 10.1109/ICSSS58085.2023.10407576.

- [60] P. Antonopoulos *et al.*, "Constant time recovery in Azure SQL database," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 2143–2154, Aug. 2019, doi: 10.14778/3352063.3352131.
- [61] S. Das *et al.*, "Automatically Indexing Millions of Databases in Microsoft Azure SQL Database," in *Proceedings of the 2019 International Conference on Management of Data*, New York, NY, USA: ACM, Jun. 2019, pp. 666–679. doi: 10.1145/3299869.3314035.
- [62] Y. Zhu *et al.*, "Towards Building Autonomous Data Services on Azure," in *Companion of the 2023 International Conference on Management of Data*, New York, NY, USA: ACM, Jun. 2023, pp. 217–224. doi: 10.1145/3555041.3589674.
- [63] S. Mathew, "Overview of Amazon Web Services," 2014.
- [64] V. B. -, A. K. -, A. M. S. -, and M. Kr. R. -, "Fortifying the Cloud: Unveiling the Next-Generation Security Model of AWS," *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, vol. 11, no. 3, Jun. 2023, doi: 10.37082/IJIRMPS.v11.i3.230230.
- [65] U. De Montemorelos, "ARQUITECTURA DE BACK END CON AMAZON WEB SERVICES (AWS) PARA SISTEMAS ESCOLARES."
- [66] T. Dancheva, U. Alonso, and M. Barton, "Cloud benchmarking and performance analysis of an HPC application in Amazon EC2," *Cluster Comput*, Jun. 2023, doi: 10.1007/s10586-023-04060-4.
- [67] Z. Hasani, M. Kon-Popovska, and G. Velinov, "Lambda Architecture for Real Time Big Data Analytic." [Online]. Available: <http://storm-project.net/>
- [68] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff, "How Amazon web services uses formal methods," *Commun ACM*, vol. 58, no. 4, pp. 66–73, Mar. 2015, doi: 10.1145/2699417.
- [69] S. Candidate, L. De Russis, and A. Baroso, "Development of a Web Application for Project Management using Amazon Web Services and Microfrontends."
- [70] H. Le Minh, "Migration project with Serverless Frame-work and Amazon Web Services API Technology and Communication 2020 2 ACKNOWLEDGMENT."

- [71] Y. Ageeva, "Development of a serverless web-application for large scale IoT platform administration," 2019.