# Modelado y Aprovisionamiento Automático de Infraestructura de Microservicios mediante Ingeniería Dirigida por Modelos

.

**Alberto Cortez [1,2], Carlos Martinez [1,2] , Claudia Naveda [1,2], Mariana Brachetta [1]**

**Pablo Peña [1] , Raúl Moralejo [1], Alejandro Vazquez [1]**

[1] Laboratorio de Auditoria y Seguridad de TI _ UTN Facultad Regional Mendoza – Mendoza -Argentina

[2] Instituto de Informática de la Facultad de Ingeniería - Universidad de Mendoza - Mendoza- Argentina

cortezalberto@gmail.com, carlos.martinez@frm.utn.edu.ar,

claudialaboral@gmail.com, mariana.brachetta@docentes.frm.utn.edu.ar, pablorpena909@gmail.com, rmoralejo@frm.utn.edu.ar , avazquez@frm.utn.edu.ar

**Abstract**

Within the framework of infrastructure as code (IAC), the deployment and control of a complex microservices-oriented application over a heterogeneous set of cloud providers. Repetitive and error-prone manual tasks are generated, this requires highly trained personnel to configure the entire environment effectively. Suppliers expose their services according to independent specifications, incurring a lack of portability and interoperability, converging on the problem that involves the restricted or proprietary use of a technology, solution or service developed by a supplier.

To address this problem, this article uses Model Driven Engineering and the Infrastructure as Code paradigm. A domain-specific language and a tool are proposed that allow graphically modeling the microservices infrastructure for deployment in the cloud. From the model, cross-platform scripts can be generated as a solution within the framework of infrastructure as code.

In summary, the objective of this work is to offer a solution for cloud infrastructure management, promoting portability and interoperability through an Infrastructure as Code approach supported by a Domain Specific Language and a modeling tool.

**Keywords:** Model-Driven Software Engineering, Transformations, DevOps, IaaS, IaC, Microservices, CTS.

**Resumen**
En el marco de la infraestructura como código (IAC), el despliegue y control de una aplicación compleja orientada a microservicios sobre un conjunto heterogéneo de proveedores en la nube. Se generan tareas manuales repetitivas y tendientes a errores, esto requiere personal altamente capacitado para configurar todo el entorno de manera efectiva. Los proveedores exponen sus servicios de acuerdo con especificaciones independientes incurriendo en una falta de portabilidad e interoperabilidad convergiendo en la problemática que implica el uso restringido o propietario de una tecnología, solución o servicio desarrollado por un proveedor.
Para abordar este problema, en este artículo se utiliza la Ingeniería Dirigida por Modelos y el paradigma de Infraestructura como Código. Se propone un lenguaje específico de dominio y una herramienta que permite modelar gráficamente la infraestructura de microservicios para su despliegue en la nube. A partir del modelo se pueden generar scripts multiplataforma como solución en el marco de la infraestructura como código.
En resumen, el objetivo de este trabajo es ofrecer una solución para la gestión de la infraestructura en la nube, promoviendo la portabilidad y la interoperabilidad a través de un enfoque de Infraestructura como Código respaldado por un Lenguaje Específico de Dominio y una herramienta de modelado.

**Palabras Claves:** Ingeniería de Software Dirigida por Modelos, Transformaciones, DevOps, IaaS, IaC, Microservicios, CTS.

## 1. Introducción

La Ingeniería Dirigida por Modelos (MDE, por sus siglas en inglés)(Schmidt,2006), es un paradigma de desarrollo de software que utiliza modelos para la generación de código y otros artefactos (Swithinbank et al., s.f.). Estos modelos pueden crearse con lenguajes de modelado de propósito general, por ejemplo, UML (OMG, s.f.) o con un lenguaje específico de dominio (DSL, por sus siglas en inglés) (Brambilla et al.,2017). Incorpora al proceso de producción de software la abstracción y el formalismo necesario, para automatizar y optimizar tareas críticas del proceso de desarrollo. De esta forma, MDE permite mejorar la productividad, la portabilidad, la interoperabilidad, la escalabilidad, la reutilización de código y el mantenimiento de los sistemas.

La Ingeniería de Software Dirigida por Modelos se puede utilizar para proveer soporte al proceso de aprovisionamiento de infraestructura como código en la nube y microservicios aplicando las mejores prácticas de DevOps.

El área de DevOps (Cois et al.,2017), está conformada por desarrolladores y personal de operaciones, que necesitan definir, actualizar y ejecutar los recursos de

infraestructura en diferentes proveedores de servicios en la nube. La Infraestructura como servicio (IaaS, por sus siglas en inglés) (Microsoft Azure,2023), es un tipo de servicio de informática en la nube que ofrece recursos esenciales de proceso, almacenamiento y redes a petición que son de pago por uso . La Infraestructura como Código (IaC, por sus siglas en inglés ) (Morris,2016) , es el proceso de automatizar los cambios en la infraestructura a través de código, para lograr escalabilidad, fiabilidad y seguridad.

El problema que los clientes de las plataformas cloud deben enfrentar es el despliegue y control de una aplicación compleja sobre un conjunto heterogéneo de proveedores. Los proveedores de servicios IaaS ofrecen diferentes tipos de infraestructura con diversidad de lenguajes de scripting para: definir, actualizar y ejecutar la infraestructura en la nube. Un ejemplo de esta problemática es la diferencia entre Amazon Cloud Formation y Azure Resource Manager, dos lenguajes cuyo principal uso es el mismo pero difieren en su sintaxis.

Los proveedores exponen sus servicios de acuerdo con especificaciones heterogéneas incurriendo en una falta de portabilidad e interoperabilidad convergiendo en la problemática que implica el uso restringido o propietario de una tecnología, solución o servicio desarrollado por un proveedor.

La integración continua y el aprovisionamiento de la infraestructura en la nube, provoca tareas repetitivas. Además se necesita personal con un alto grado de capacitación para configurar todo el entorno.

Este proyecto de investigación tiene como objetivo optimizar el proceso de aprovisionamiento de infraestructura en la nube mediante el diseño e implementación de un artefacto que permita reducir riesgos y mejorar la calidad en el proceso productivo del software.

Para resolver el problema planteado se propone proveer soporte para la gestión de herramientas DevOps, a través de la definición de un Lenguaje Específico de Dominio basado en el concepto de Infraestructura como Código, y una herramienta que apoya este lenguaje que permite modelar el estado final de un aprovisionamiento infraestructura en la nube como así también el modelado de los microservicios orquestados, generando el aprovisionamiento de scripts multiplataforma como solución guiada por la infraestructura como código. La herramienta propuesta optimiza y facilita el proceso de desarrollo en un marco de calidad en base al desarrollo de software dirigido por modelos.

El aporte de esta propuesta es Accelerate, una herramienta de modelado de infraestructura para el aprovisionamiento de recursos, la orquestación y el despliegue microservicios en la nube, que pretende abstraer la complejidad de trabajar con diferentes herramientas DevOps a través de un dominio específico y proveer una configuración inicial parametrizable, que pueda ser utilizada como un marco de trabajo que guíe tanto el desarrollo como el despliegue del software.

El documento está organizado de la siguiente manera. La sección 2 describe los trabajos relacionados, continuando la sección 3 se presenta una herramienta de modelado de infraestructura: Accelerate, posteriormente en la sección 4 se presenta el funcionamiento de Accelerate, por último en la sección 5 se establecen las conclusiones y los trabajos futuros.

## 2. Trabajos relacionados

En la actualidad, existen varias herramientas para gestionar el aprovisionamiento de infraestructura. Estas utilizan scripts para definir el estado final de la infraestructura en la nube. Sin embargo, la gestión de estas herramientas mediante distintos lenguajes de scripts implica para los DevOps una tarea manual que requiere mucho tiempo y es propensa a introducir errores. Además, la integración infraestructura-microservicio se convierte en una ardua tarea cuya complejidad crece a la par de la arquitectura del software que se pretende desplegar.

Existen trabajos de investigación relacionados que abordan los problemas de la Infraestructura como Código mediante el desarrollo dirigido por Modelos (MDD, por sus siglas en inglés). Podemos destacar a: Borisova, Sandobalín y Brabra.

En Borisova et al., 2020) se analiza la aplicabilidad del estándar TOSCA (Lauwers et al., s.f.), para unificar el despliegue y la orquestación de recursos provistos por la virtualización basada en la nube y también usando contenedores.

En (Sandobalín et al., 2017) se presenta una herramienta para modelar el estado final de aprovisionamiento de la infraestructura en la nube a través de un DSL con una notación gráfica.

En (Brabra et al., 2019) propone un enfoque de orquestación dirigido por modelos, que utiliza TOSCA para describir artefactos de recursos en la nube. Utilizan una técnica de transformación en el marco de la ingeniería dirigida por modelos (MDE).

La diferencia fundamental de las propuestas (Borisova et al., 2020), (Sandobalín et al., 2017) y (Brabra et al., 2019) con nuestro enfoque, está en el metamodelo propuesto y la notación gráfica con Sirius (Eclipse Foundation, 2020), que representa la sintaxis concreta. Los elementos considerados en el metamodelo permiten el diseño de la notación gráfica que contempla el aprovisionamiento con contenedores utilizando Kubernetes Engine y otros proveedores de servicios como: Google y Amazon.

Esto le permite al DevOps contar con una herramienta gráfica fácil de usar y flexible. Además se brinda una solución que abstrae la complejidad de especificar los recursos de infraestructura de diversos proveedores de servicios, así como la automatización de la generación de scripts para el aprovisionamiento y configuración de la infraestructura en la nube.

Nuestra propuesta permite el modelado de la infraestructura teniendo en cuenta las aplicaciones que se van a desplegar, como las clases y las rutas de cada microservicio. Se integra no sólo la infraestructura, sino también los microservicios y el código, permitiendo un despliegue ya integrado.

## 3. Herramienta de modelado de infraestructura: Accelerate

En esta sección presentamos Accelerate, una herramienta de software que permite modelar la infraestructura de un sistema basado en microservicios para su posterior despliegue en la nube. Este modelado se realiza utilizando un lenguaje específico de dominio a través de una interfaz gráfica. Además, la herramienta permite generar el código para el aprovisionamiento automático en la nube utilizando el modelo como entrada. Esta generación permite utilizar el paradigma de Infraestructura como Código (IaC) (Morris, 2016), lo que permite ahorrar tiempo, realizar control de versiones de la

infraestructura, conseguir mayor flexibilidad, entre otras ventajas.

### 3.1 Arquitectura

Accelerate tiene como objetivo apoyar al equipo en términos de modelado de la infraestructura utilizando una herramienta gráfica, donde el modelo no se utiliza con fines meramente representativos, sino que permite la obtención de código de la infraestructura diagramada. La Figura 1 representa el metamodelo de infraestructura, donde podemos encontrar clases abstractas en color gris y clases concretas en color amarillo con el siguiente significado:

- ProvidersDiagram: Clase raíz del metamodelo, que representa a todo el diagrama.
- Provider: Ambiente de un proveedor en la nube. Esta clase abstracta a su vez está relacionada a un segundo metamodelo, donde la clase Provider se convierte en contenedora de elementos de la clase Microservice, representando una instancia puntual de un microservicio. De esta manera, la clase Provider representa un ambiente de microservicios que será desplegado en un proveedor en la nube.
- GoogleProvider: Implementación de la clase Provider en Google Cloud, donde se almacena el identificador del proyecto y la región de despliegue.
- Configuration: Configuración de un aspecto puntual del ambiente en la nube.
- KubernetesConfiguration: Configuración de Kubernetes de un proveedor en la nube.
- GoogleKubernetesConfiguration: Implementación de una configuración de Kubernetes para un proveedor en la nube, donde se almacena todo lo relacionado al tamaño del clúster de Kubernetes subyacente al ambiente de microservicios.

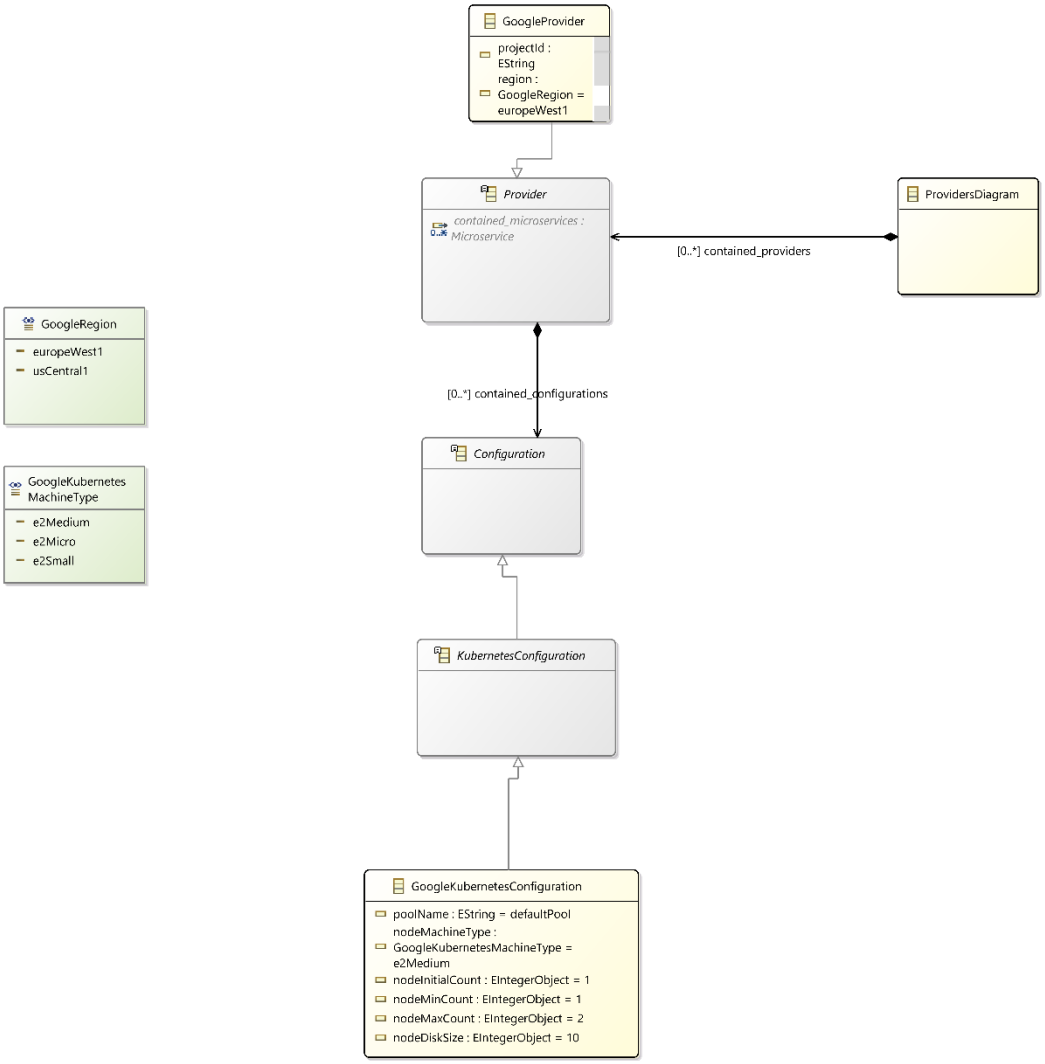


**Figura 1: Diagram Provider**

La Figura 2, muestra el funcionamiento general de Accelerate, una herramienta con una arquitectura de plugins desarrollados sobre Eclipse Modeling Framework (Eclipse Foundation, 2020).

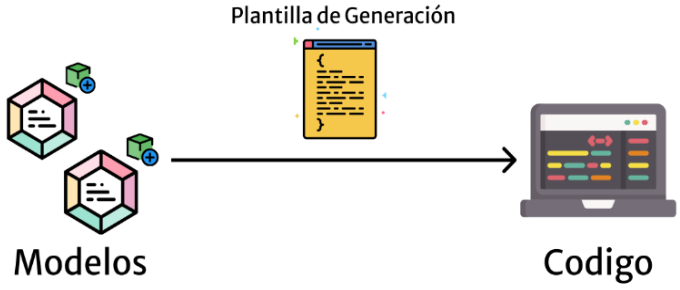


**Figura 2: Modelado a código**

La arquitectura tiene principalmente dos componentes: 1) Componente de modelado, que permite diagramar un modelo de infraestructura y 2) Componente de generación automática, que recibe el modelo creado y genera los scripts de aprovisionamiento de forma automática. A continuación, se describe cada uno de los componentes:

Componente de modelado: Representa la sintaxis concreta del lenguaje específico de dominio y surge a partir de la sintaxis abstracta, es decir, los metamodelos. Su interfaz gráfica fue creada utilizando Sirius junto con Graphical Modeling Framework (GMF) (Eclipse Foundation, 2020). Con este componente, podemos crear múltiples ambientes de microservicios y configurarlos de forma interactiva, se puede ver un ejemplo en la Figura 3.

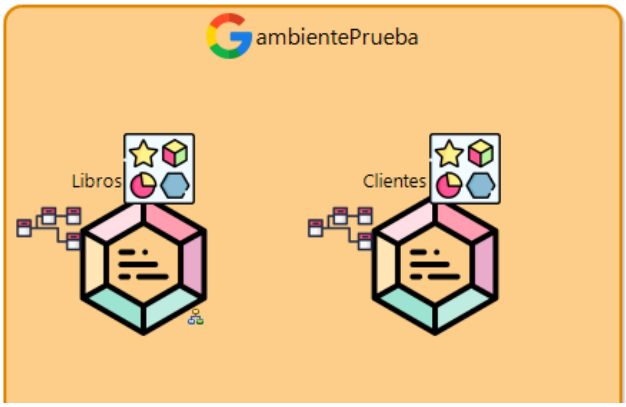


**Figura 3: Ambientes de Microservicios**

A su vez, también permite disparar la generación automática de la infraestructura, es decir, ejecutar el segundo componente a través del elemento Generate Code, como se muestra en la Figura 4.

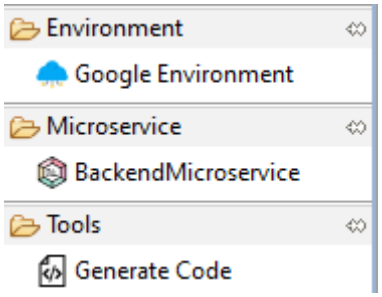


**Figura 4: Menú de Generación**

Componente de generación automática: Realiza las transformaciones de Modelo a Texto (M2T) (Eclipse Foundation, 2020) a partir del modelo creado con el primer componente, generando los scripts para el aprovisionamiento automático de la infraestructura. Este componente fue desarrollado con Acceleo como motor de generación de código y el lenguaje de restricciones de objetos (OCL , por sus siglas en inglés) como lenguaje para definir las plantillas de generación automática (Eclipse Foundation, 2020).

## 4. Funcionamiento de Accelerate

En esta sección, se muestra un fragmento de un caso real para demostrar el funcionamiento de la herramienta Accelerate en términos de modelado y generación automática de infraestructura. Este caso plantea modelar y desplegar en Google Cloud Platform (GCP) (Morris, 2016) una aplicación de software básica para la gestión de libros y clientes de una biblioteca. Este sistema está compuesto inicialmente por dos microservicios, uno para gestionar los clientes y otro para gestionar los libros. Es importante aclarar que estos microservicios no estarán aislados, ya que los clientes alquilan y devuelven libros, por lo que se podría establecer una relación entre ambos microservicios como se muestra en la Figura 5.

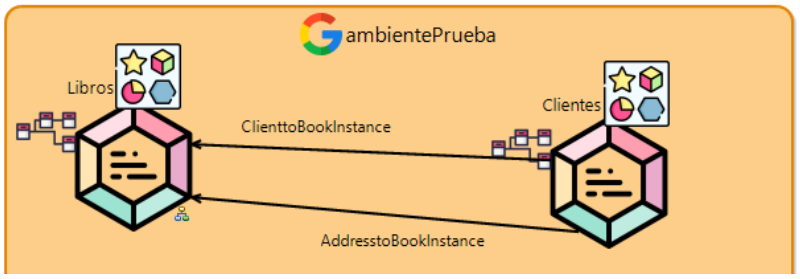


**Figura 5: Despliegue en Google Plataform**

Otro aspecto importante es la capacidad de cómputo subyacente que se desea obtener a través de GCP y que puede modelarse a través de ventanas emergentes en múltiples aspectos:

- **Capacidad de cómputo del clúster**: Al incluir dos microservicios en un clúster de Google Cloud Platform, asumimos que este será un clúster que utilice la implementación de Kubernetes que provee Google, GKE (Google Kubernetes Engine). A su vez, esta implementación requiere información respecto a su pool de nodos como se muestra en la Figura 6.

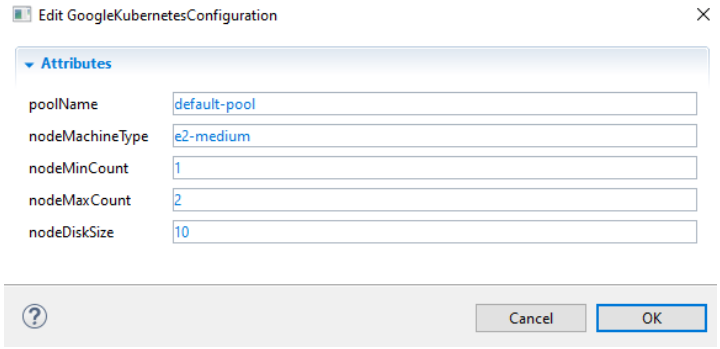


**Figura 6: Capacidad de Cluster**

- **Escalado horizontal de los servicios:** Ya que los microservicios serán desplegados en un clúster de Kubernetes, se debe definir algunos aspectos como

la cantidad de réplicas de cada servicio, los recursos asignados, etc. Este aspecto se modela a través de una ventana emergente como la de la Figura 7.

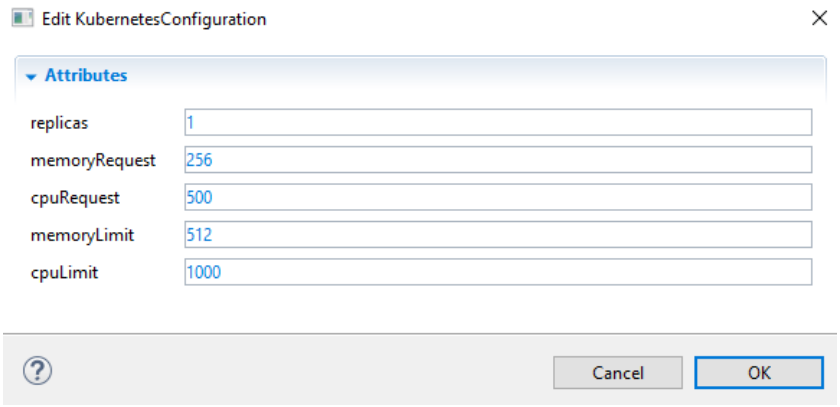


**Figura 7: Capacidad de escalado**

- **Inyección de variables de entorno:** Otro aspecto interesante de la generación automática es la posibilidad de inyectar variables de entorno al microservicio a través de los manifiestos de Kubernetes. De esta manera, podemos modelar variables que luego puedan ser accedidas por el mismo servicio a través de las variables de entorno. Un ejemplo puntual sería definir que el microservicio accede a una base de datos MongoDB, la cual puede ser accedida con un formato de URL e inyectada como variable de entorno (MongoDB, 2023). El modelado de este tipo de capacidades se muestra en la Figura 8 y esa información es mapeada a una URL con formato: **mongodb + srv://<user>:<pass> @<host>/<databaseName>**

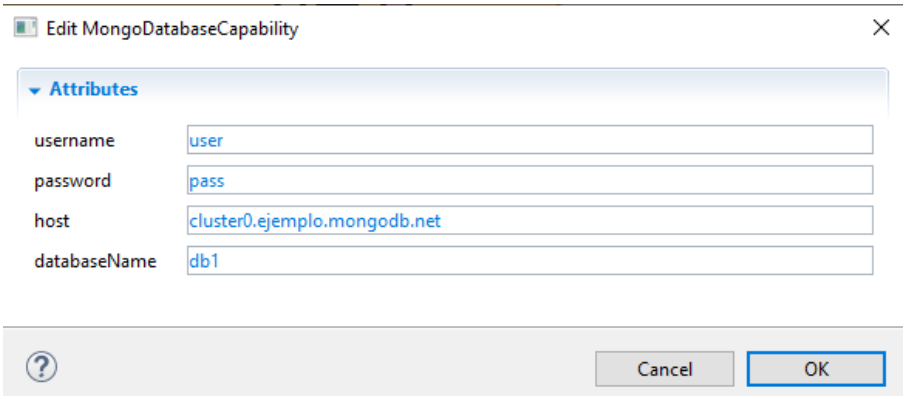


**Figura 8: Credenciales de Seguridad**

Este modelo realizado de forma gráfica luego pasa a través del motor Acceleo, que aplica una transformación M2T y genera de forma automática los archivos de tipo yaml relacionados con la infraestructura. Esta generación automática puede ser segmentada en dos aspectos:

**Aprovisionamiento de infraestructura:** Este aspecto refiere a los archivos necesarios para obtener los recursos de despliegue, es decir, aprovisionarse de un clúster de GKE que tenga las características modeladas. Este aprovisionamiento involucra a Terraform,

una herramienta de infraestructura como Código desarrollada por HashiCorp (Morris, 2016). Los archivos son generados con la sintaxis de HCL, lenguaje de configuración de Hashicorp, como se muestra en la Figura 9.

```
module "gke" {
  source                 = "terraform-google-modules/kubernetes-engine/google//modules/private-cluster"
  project_id             = var.project_id
  name                   = "${var.cluster_name}-${var.env_name}"
  region                 = "southamerica-west1"
  zones                  = ["southamerica-west1-a"]
  network                = module.gcp-network.network_name
  subnetwork             = module.gcp-network.subnets_names[0]
  ip_range_pods          = var.ip_range_pods_name
  ip_range_services      = var.ip_range_services_name
  regional               = true
  node_pools = [
    {
      name                      = "default-pool"
      machine_type              = "e2-medium"
      node_locations            = "southamerica-west1-a"
      min_count                 = 1
      initial_node_count        = 1
      max_count                 = 2
      disk_size_gb              = 10
    },
  ]
}
```

**Figura 9. Fragmento del archivo de Terraform**

**Orquestación de microservicios:** El segundo aspecto hace referencia a la orquestación de los microservicios dentro del clúster de GKE. Al ser este un clúster de Kubernetes, es necesario obtener manifiestos de configuración que permitan su despliegue, su gestión y su escalado en el clúster mencionado. Para ello, se generan de forma automática los archivos de Deployments para la creación de los contenedores, los de Services para su intercomunicación y un Ingress para la distribución del tráfico como se muestra en la Figura 10.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-backend
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
  - http:
      paths:
      - path: /clientes(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: clientes
            port:
              number: 8000
      - path: /libros(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: libros
            port:
              number: 800
```

**Figura 10. Ejemplo de archivo Ingress para redistribuir el trafico del cluster**

Finalmente, la persona poseedora de estos archivos, solamente debe aplicar los archivos de configuración mencionados para obtener la infraestructura y desplegar los microservicios en la misma.

## 5. Conclusiones y Trabajo Futuro

Este artículo se enfoca en el problema de gestionar la infraestructura en la nube mediante la aplicación de la Ingeniería Dirigida por Modelos y el enfoque de la Infraestructura como Código.

En el contexto de la Infraestructura como Código (IAC), se presentan desafíos al desplegar y administrar una aplicación compleja en una variedad de proveedores de servicios en la nube. Esto conlleva la realización de tareas manuales repetitivas y propensas a errores, demandando personal altamente capacitado para configurar eficazmente el entorno. Los proveedores ofrecen servicios siguiendo especificaciones individuales, lo que resulta en falta de portabilidad e interoperabilidad. Esto lleva a la restricción o dependencia de tecnologías, soluciones o servicios propietarios ofrecidos por un proveedor específico.

Como propuesta de solución a los problemas planteados, se presenta la herramienta Accelerate, compuesta por sus dos componentes principales: el Componente de Modelado y el Componente de Generación Automática.

El Componente de Modelado, ofrece una interfaz gráfica robusta desarrollada con Sirius y Graphical Modeling Framework. Esto permite la creación interactiva de modelos de infraestructura, facilitando la configuración y el diseño de entornos de microservicios.

Por otro lado, el Componente de Generación Automática, impulsado por Acceleo y el lenguaje OCL, realiza transformaciones de Modelo a Texto (M2T) basadas en el modelo creado con el Componente de Modelado.

Esta generación automática engloba tanto el aprovisionamiento de la infraestructura, utilizando Terraform, como la orquestación de los microservicios en un clúster de Kubernetes.

En este trabajo se demuestra la eficacia de la herramienta Accelerate, al simplificar y mejorar la gestión de infraestructura y microservicios en la nube. La integración de modelado interactivo, generación automática y enfoque de Infraestructura como Código ofrece una solución robusta y eficiente para abordar los desafíos actuales en el despliegue de aplicaciones en entornos en la nube. La facilidad de uso y la automatización resultante aseguran una implementación consistente y de alta calidad, al tiempo que reducen la complejidad y el tiempo de implementación.

Este artículo demuestra el funcionamiento de Accelerate, una aplicación práctica de la Ingeniería Dirigida por Modelos a la generación automática no sólo de código, sino también de infraestructura, un área poco explorada pero muy prometedora.

El uso masivo de este tipo de herramientas permitirá a los equipos de trabajo ahorrar tiempo, ahorrar dinero y crear mejores soluciones, ya que evitará aquellas tareas repetitivas mediante la generación automática, logrando sentar bases fundadas en las mejores prácticas de desarrollo y así guiar al equipo hacia productos de alta calidad.

Una de las características más importantes de Accelerate es su facilidad de uso, ya que ofrece una interfaz gráfica sencilla pero completa, que podrá ser utilizada tanto por personas que recién comienzan a desarrollar arquitecturas de microservicios en la nube como así también por profesionales expertos en el tema. Esta interfaz gráfica luego traduce el modelado a código e infraestructura, aportando un alto valor añadido con tan solo un click.

Ahora bien, es evidente la utilidad de Accelerate, pero también es importante pensar en aquellas mejoras que pueden ser realizadas en integración con otras tecnologías. Una herramienta de este calibre, podría integrarse con tecnologías disruptivas como la inteligencia artificial generativa, mejorando la generación de código en pos de conseguir una sola herramienta que acompañe todo el ciclo de desarrollo y despliegue de un producto de software.

## 6. Referencias

Schmidt, D.C. (2006). Model-driven engineering. COMPUTER-IEEE COMPUTER SOCIETY- 39(2), 25.

Swithinbank, P., Chessell, M., Gardner, T., Griffin, C., Man, J., Wylie, H., & Yusuf, L. (sin fecha). Patterns: Model-Driven Development Using IBM Rational Software Architect. Disponible en ibm.com/redbooks.

OMG. (2010). Unified Modeling Language (UML), versión 2.4.1. Recuperado de http://www.omg.org/spec/UML/2.4.1/

Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-Driven Software Engineering in Practice (Segunda edición). Morgan & Claypool Publisher.

Cois, C.A., Yankel, J., & Connell, A. (2014). Modern devops: Optimizing software development through effective system interactions. En IPCC. IEEE, pp. 1-7.

Microsoft Azure.(2023). ¿Qué es IaaS? Recuperado de https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-is-iaas/

Morris, K. (2016). Infrastructure As Code: Managing Servers in the Cloud. O'Reilly & Associates Incorporated.

Borisova, A., Shvetcova, V., & Borisenko, O. (2020). Adaptation of the TOSCA standard model for the Kubernetes container environment. Proceedings - 2020 Ivannikov Memorial Workshop, IVMEM 2020, 9–14. https://doi.org/10.1109/IVMEM51402.2020.00008

Lauwers, C., & Tamburri, D. (s.f.). OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC. Recuperado de https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

Sandobalín, J., Insfran, E., Abrahão, S. (2017). Automatización del Aprovisionamiento de Infraestructura en la Nube. 11705/JCIS/2017/018

Brabra, H., Mtibaa, A., Gaaloul, W., Benatallah, B., & Gargouri, F. (2019). Model-driven orchestration for cloud resources. IEEE International Conference on Cloud Computing, CLOUD, 2019-July. https://doi.org/10.1109/CLOUD.2019.00074

Eclipse Foundation. (2020). The easiest way to get your own Modeling Tool. Recuperado de https://www.eclipse.org/sirius/

MongoDB.(2023).Release Notes. Recuperado de https://www.mongodb.com/docs/manual/release-notes/