El futuro digital es de todos

MinTIC

UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC2022

02

Sistema de Software para Ciclo 4a

Ciclo 4a:

Desarrollo de aplicaciones web

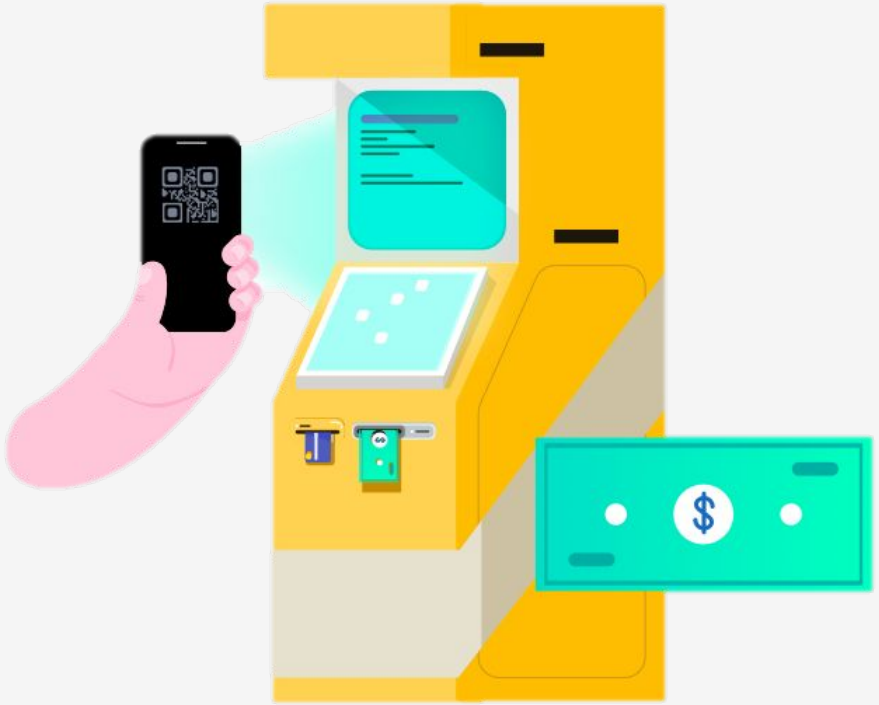
Objetivo de Aprendizaje

Analizar el **diseño** del **sistema de software** que será usado como ejemplo práctico durante el desarrollo del **Ciclo 4a**.

Introducción: Sistema a Construir

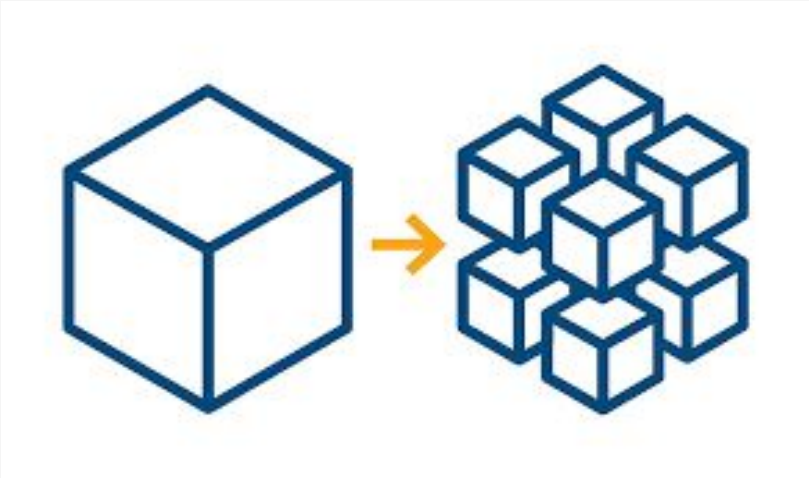
Objetivo:

Construir un sistema de software, de tipo **aplicación web**, basado en una **arquitectura de microservicios**, que le permita a un **usuario crear** una **cuenta bancaria**, **autenticarse** si ya posee una cuenta, **consultar** su **saldo**, **realizar transferencias** de dinero a otras cuentas y **consultar** su **historial** de transacciones. Además, la aplicación debe hacer uso de un mecanismo de **autenticación** basado en **tokens**.



[Imagen] ATM. (s. f.). [JPG]. MaterialUp. https://assets.materialup.com/uploads/31556231-b5aa-441f-aefa-ab668aca45ff/preview.jpg

Introducción: Microservicios



El **sistema de software** planteado requiere implementar **varias funcionalidades**, y si se quisiera trabajar con una **arquitectura monolítica**, el número de funcionalidades podría sobrecargar el componente.

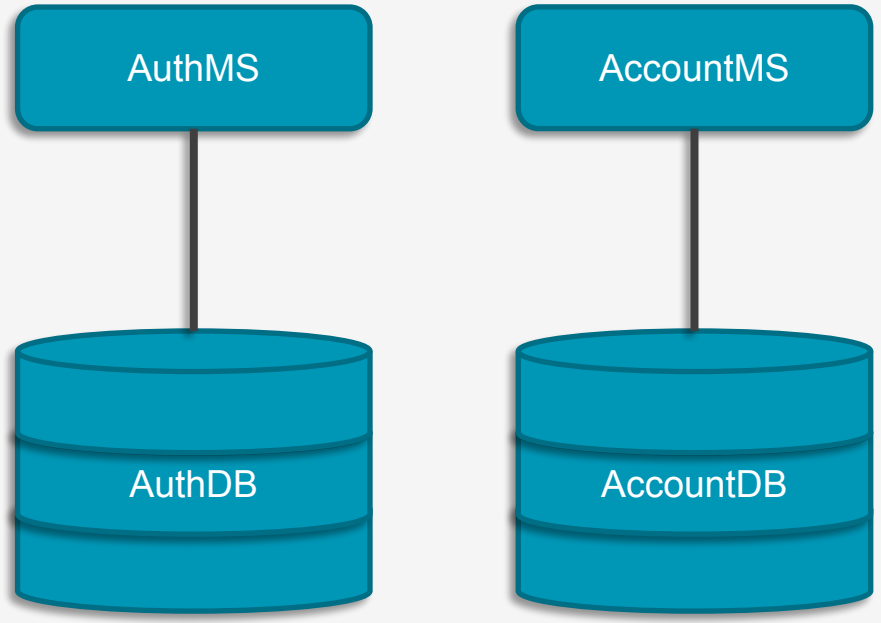
En este caso se usará una **arquitectura de microservicios**, en donde las **funcionalidades** se podrán **dividir** en grupos, teniendo en cuenta los **datos** sobre los que trabajan. Estos grupos de funcionalidades se implementarán en **microservicios** diferentes, evitando sobrecargas.

Microservices. (s. f.). [Gráfico]. https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRbLOMjwo461M72W1ui1vbSNcYS7gskTxhhyyo4SSlcE-kq0Dxfm-2U10nBHk3WhOZC8l0&usqp=CAU

Microservicios

Para el **sistema de software** planteado, se construirán 2 microservicios. Uno de ellos se encargará de implementar las funcionalidades que trabajan sobre los **datos** de **autenticación** del usuario (**AuthMs**) y el otro se encargará de implementar las funcionalidades que trabajan sobre los **datos** de la **cuenta bancaria** del usuario (**AccountMS**)

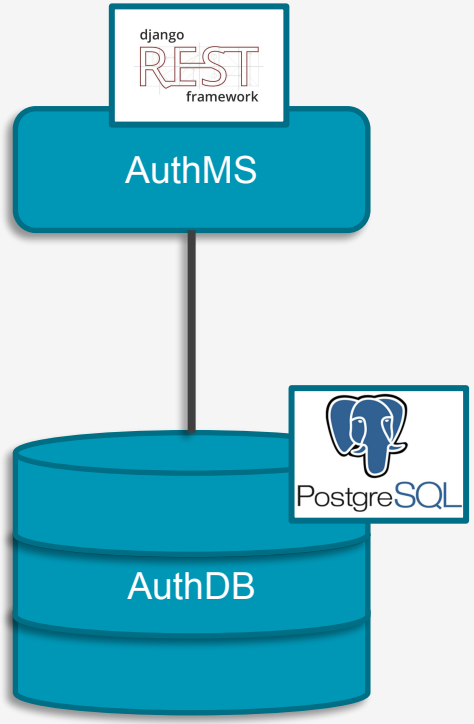
Cada uno de los microservicios tendrá una **base de datos** que le permitirá guardar los datos necesarios, asegurando así la **descentralización de los datos**.



Microservicio AuthMS

AuthMS será desarrollado en el lenguaje de programación **Python**, usando el framework **Django REST** y para asegurar la persistencia de sus datos se contará una base de datos relacional **PostgreSQL**.

En este **microservicio** se trabajará sobre los **datos** de **autenticación** del usuario, por ello se implementarán las **funcionalidades** de **registro** e **inicio** de **sesión**. Para todo este proceso se hará uso de un sistema de autenticación basado en **JWT** (**JSON Web Tokens**).

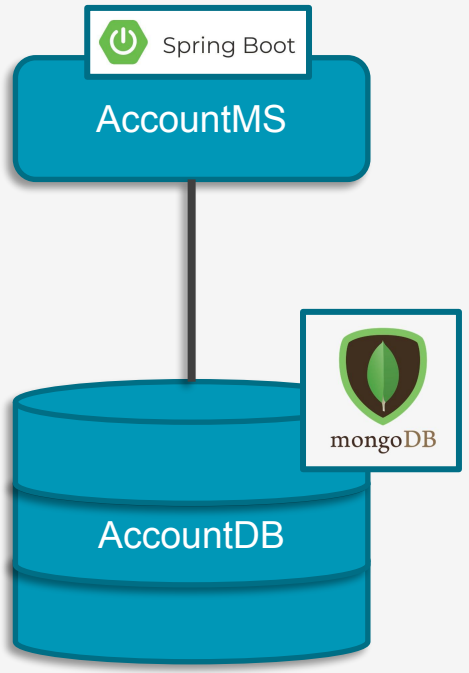


Logo PostgreSQL. (s. f.). [Ilustración]. https://cdn.holistics.io/landing/databases/postgresql.png
Logo DjangoRest. (s. f.). [Ilustración]. https://ichi.pro/assets/images/max/724/1*kR89JbQQK9aAkNVyxE63pq.png

Microservicio AccountMS

AccountMS será desarrollado en el lenguaje de programación **Java**, usando el framework **SpringBoot** y para asegurar la persistencia de sus datos se contará con una base de datos NoSQL **MongoDB**.

En este **microservicio** se trabajará sobre los **datos** de la **cuenta bancaria** del usuario (saldo, transacciones, entre otros), por ello se implementarán las **funcionalidades** de **consultar saldo**, realizar **transferencias** a otra cuenta y consultar el **historial** de transacciones.

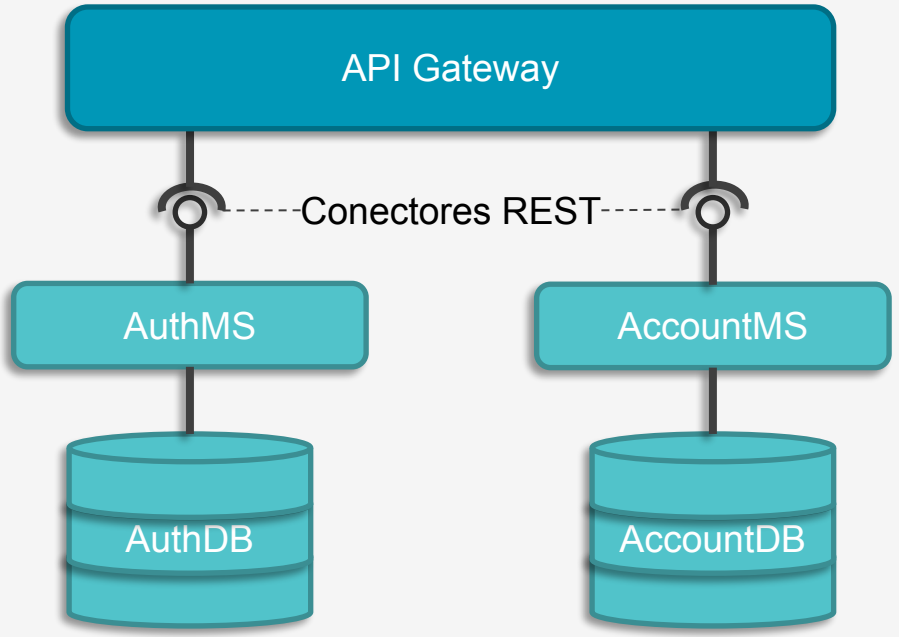


MongoDB logo. (s. f.). [Ilustración]. https://useforcode.com/wp-content/uploads/2021/04/225-2254691_9kib-354x415-unnamed-mongodb-logo-svg.jpg
SpringBoot logo. (s. f.). [Ilustración]. https://pbs.twimg.com/media/DU7GUGCV4AAf90X.jpg

API Gateway

Se contará también con un **API Gateway**, que se encargará de **recibir** solicitudes y de **redireccionarlas** a los **micro-servicios** correspondientes, para posteriormente devolver una **respuesta** al cliente.

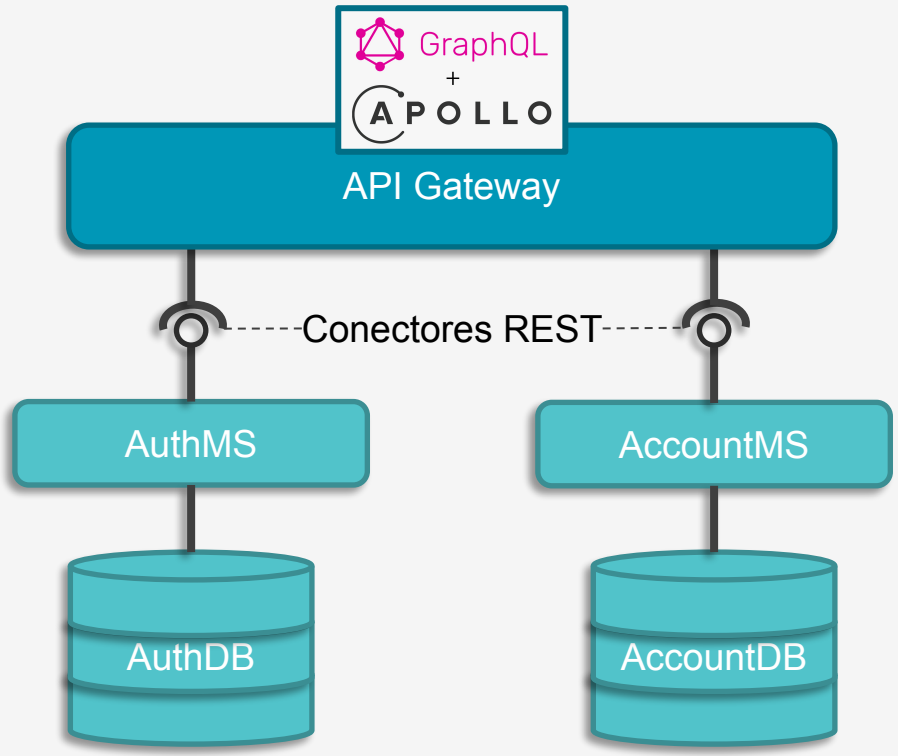
Además el API Gateway permitirá exponer **servicios** que requieren de **funcionalidades** de **ambos microservicios**, por ejemplo para realizar una transacción (**AccountMS**), es necesario que el usuario esté autenticado (**AuthMS**).



API Gateway

ELˌl **API Gateway** será desarrollado en el lenguaje de programación **JavaScript**, usando el framework **Apollo** (que trabaja sobre **Express**). Así mismo, el API Gateway expondrá una API de tipo **GraphQL,** a diferencia de los microservicios que expondrán una API de tipo **REST**.

El conector **GraphQL** brindará una serie de reglas que permiten definir de manera **rápida** y **fácil** las **peticiones** al API Gateway.

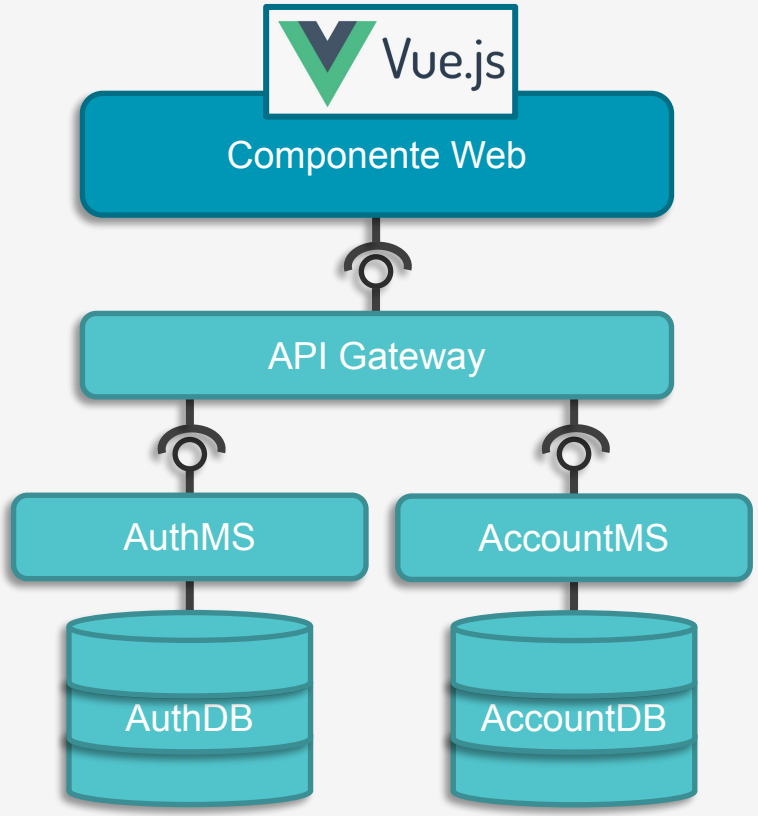


Apollo Server + GraphQL. (s. f.). [Ilustración]. https://miro.medium.com/max/3600/1*o9Fp5squKFlA6qIM1Yp10A.png

Componente Web

Por último, para que los **usuarios** puedan utilizar el sistema es necesario proveer una **interfaz**. De esta forma, se construirá un **componente web** que permitirá al cliente conectarse desde cualquier dispositivo que posea un **navegador web**, e interactuar con su **interfaz gráfica**, y de esta forma poder utilizar las **funcionalidades** de la aplicación web.

Este componente será desarrollado en el lenguaje de programación **JavaScript**, usando el framework **Vue.js**.



Logo Vue.js. (s. f.). [Ilustración]. https://www.kindpng.com/picc/m/452-4529223_vue-js-logo-png-transparent-png.png

El futuro digital es de todos

MinTIC

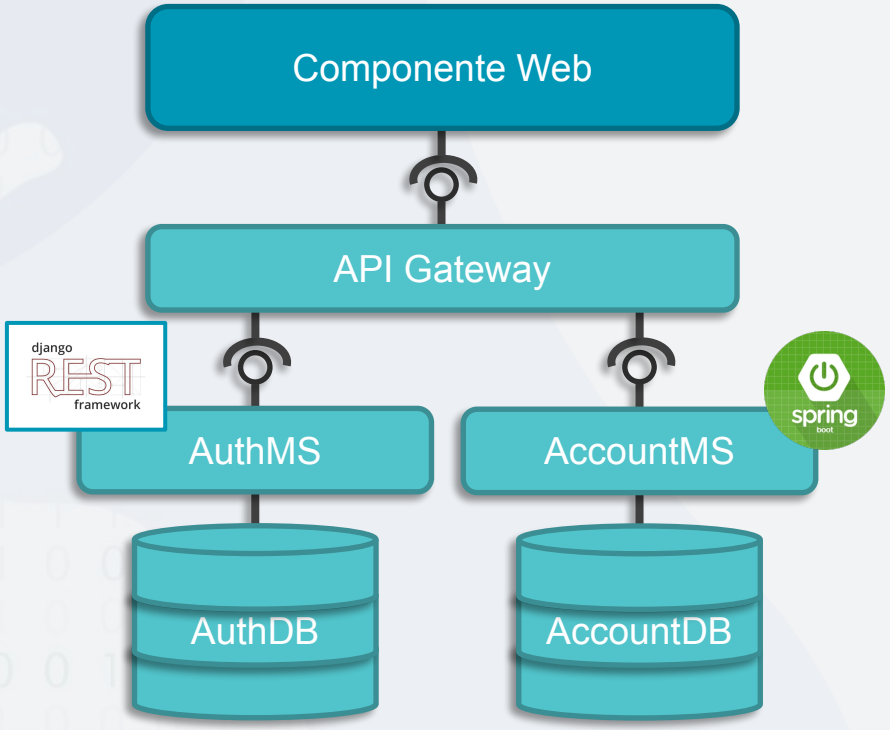
VENTAJAS

El uso de la arquitectura de microservicios en sistema de software planteado, traerá algunos bene- ficios que una arquitectura monolítica no ofrece.

UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC 2022

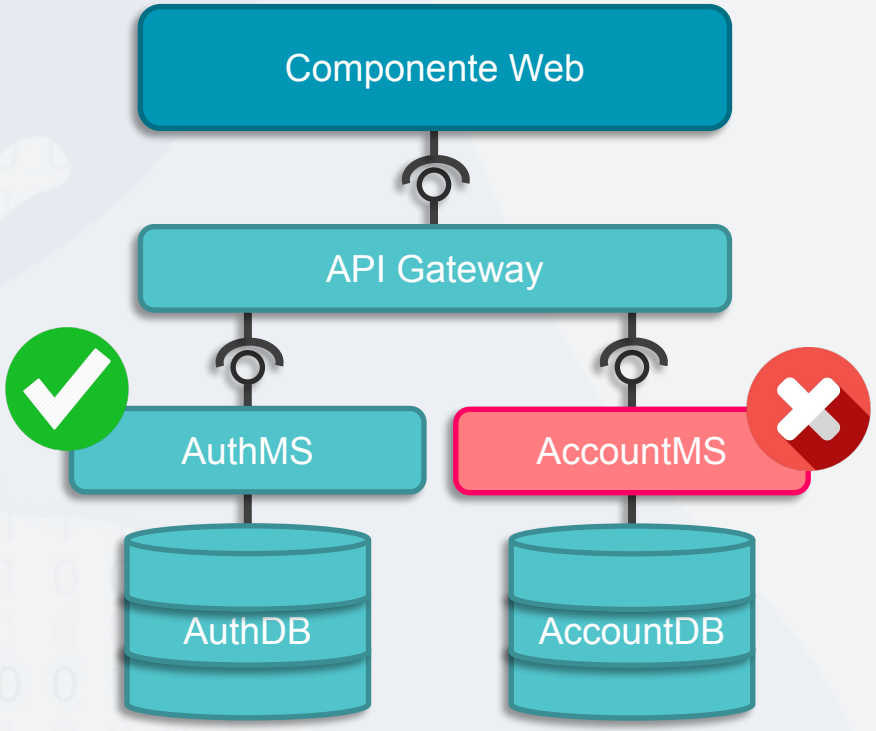
Ventajas: Heterogeneidad



En el **sistema de software** planteado se tienen dos **microservicios**, y gracias a la **heterogeneidad** se puede escoger una **tecnología** como **Django REST** que facilita la creación y manejo de usuarios **(AuthMS),** y una tecnología como **Spring-Boot** para el manejo de las transacciones **(AccountMS).**

Esto **mejora** el **rendimiento** de cada una de las **funcionalidades** ya que elimina las limitaciones de estar juntas.

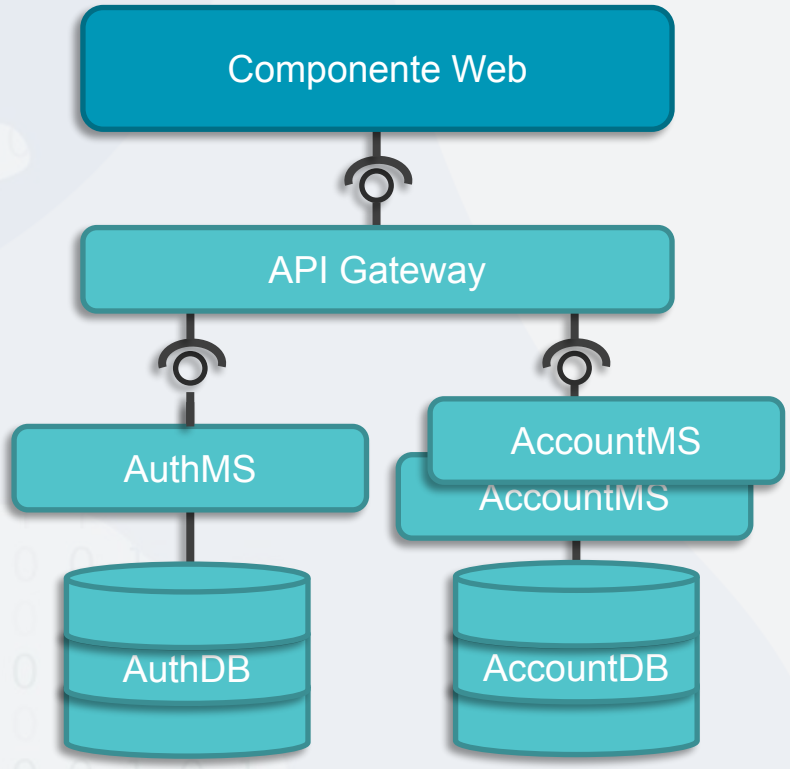
Ventajas: Tolerancia a Fallos



En el **sistema de software** planteado, si el microservicio **AccountMS falla**, como es de esperarse **no** se podrán realizar **transacciones** ya que el **microservicio no** podrá **responder** a las peticiones, pero el microservicio **AuthMS** permanecerá **intacto**, por lo cual, los **usuarios** podrán **registrarse** e **iniciar sesión** sin problemas.

En una **arquitectura monolítica** todo el sistema **fallaría**.

Ventajas: Escalabilidad



En el caso en el que el microservicio **AccountMS** requiriera recibir **más peticiones** de las esperadas y este se **saturara,** se podría **duplicar** el microservicio (escalamiento) para **responder** todas las solicitudes, lo cual resultaría **efectivo**. Mientras tanto, el microservicio **AuthMS** continuaría trabajando correctamente sin necesidad de realizar **escalamiento**.

En un **monolito** se habrían **asignado** recursos adicionales a las **funcionalidades** de **AuthMS**, cuando estas **no** los **necesitaban**.