



Platform Engineering: Impulsando eficiencia en equipos de desarrollo a través de las Plataformas de Desarrollo Interno (IDP)

Alejandra Bricio

Prácticas modernas para crear software con calidad y sabor
#SGVirtual

Platform Engineering

Transformando la eficiencia de los equipos de desarrollo



Alejandra Bricio



cloud, scripting, k8s
@alebricio

Agenda

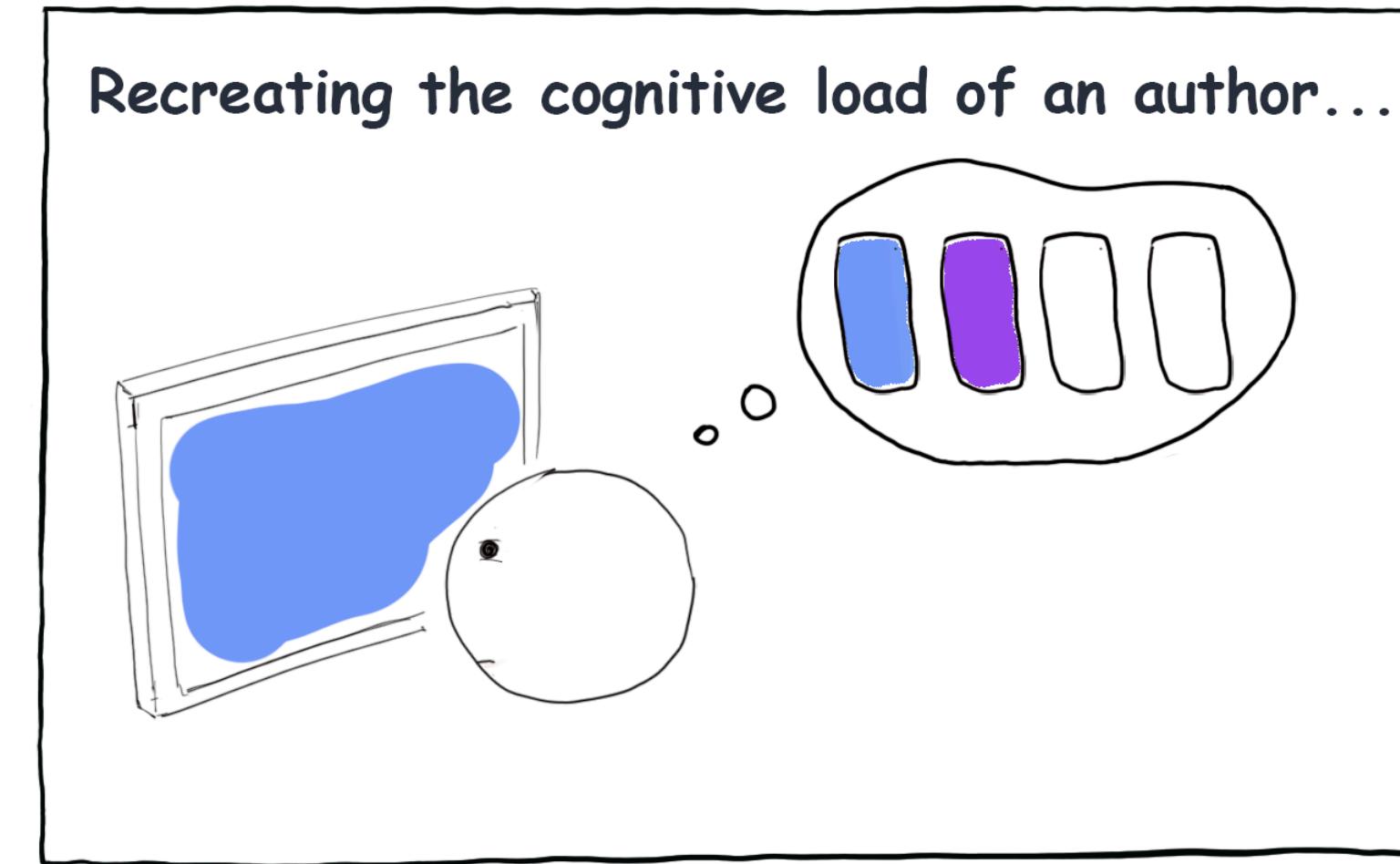
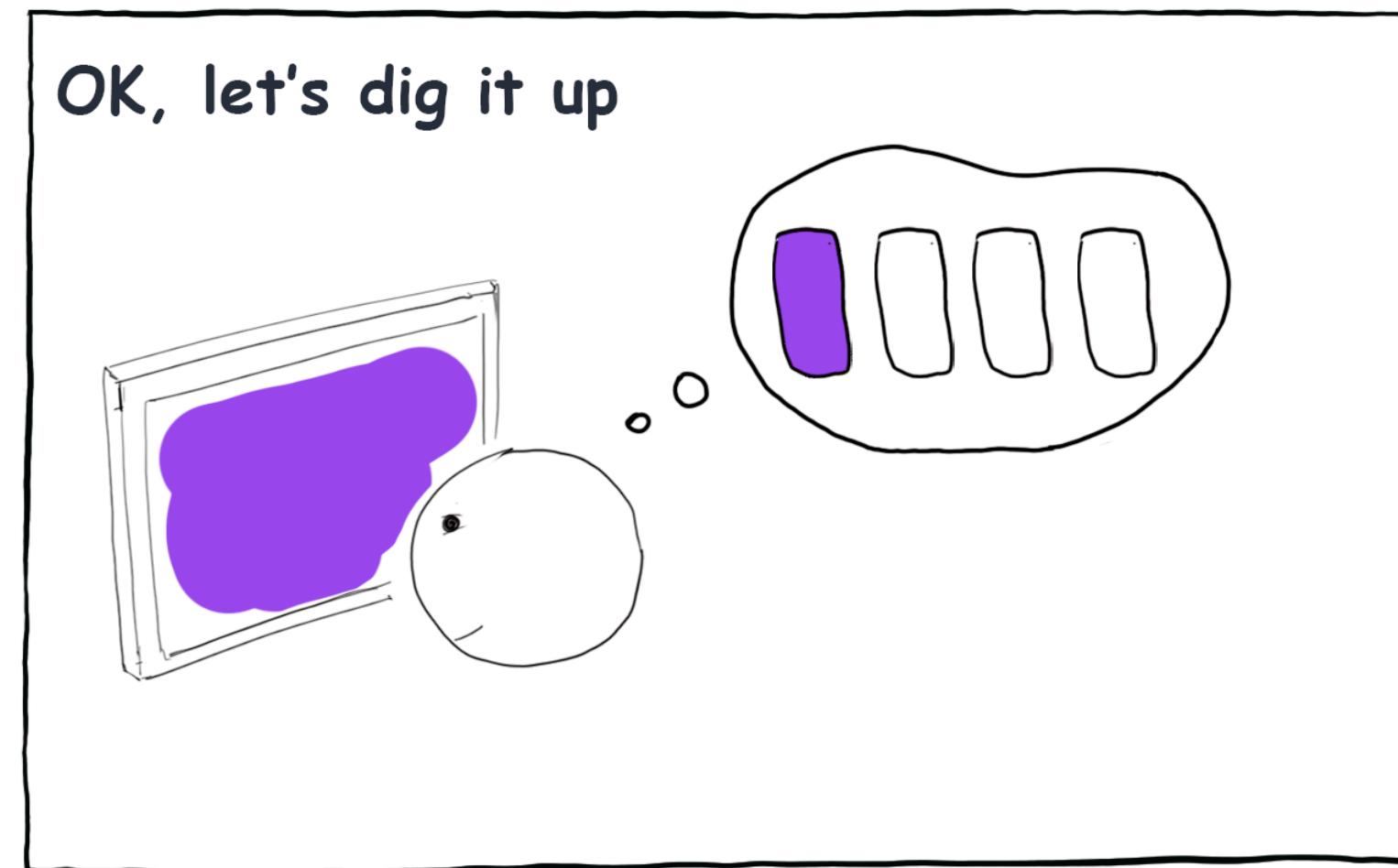
- 1. Carga Cognitiva**
- 2. Platform as a Product**
- 3. Plataformas de Desarrollo Interno (IDPs)**
- 4. Adopción**

“Paso más tiempo intentando entender cómo hacer testing, build y deploy de una aplicación **que en el desarrollo real** del código que quiero añadir.”

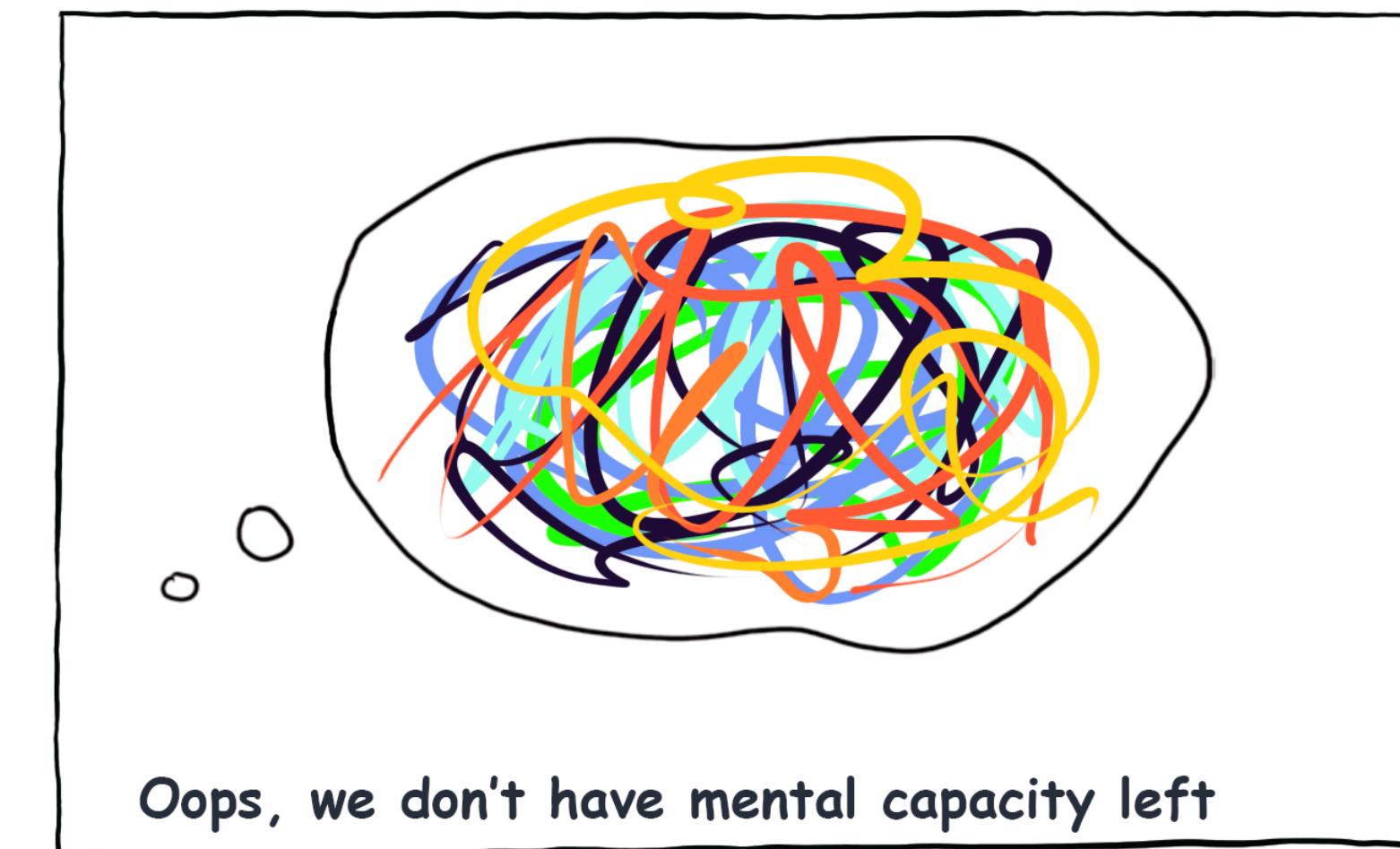
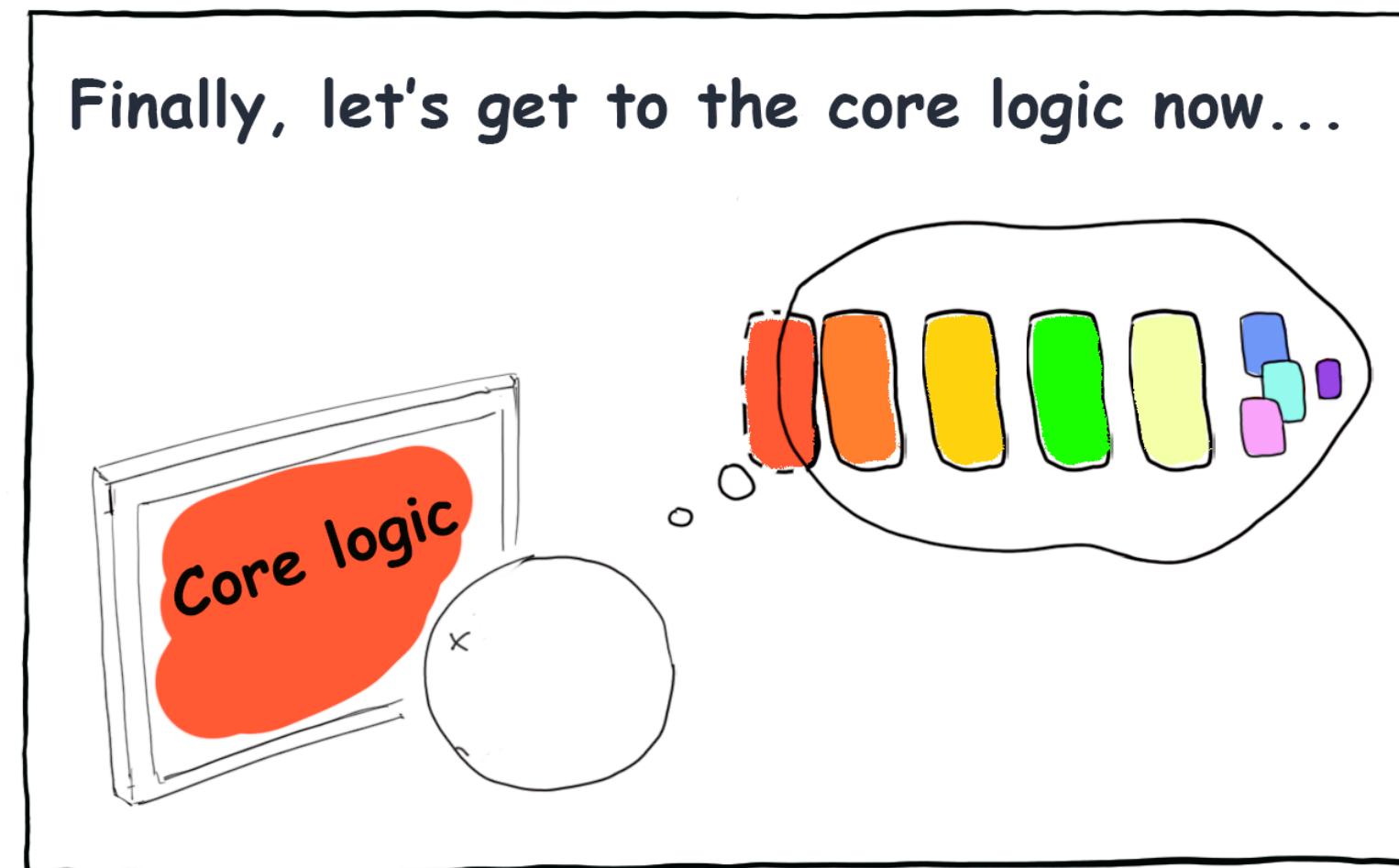
- Developer Anónimo

iPlatform Engineering al rescate!

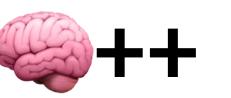




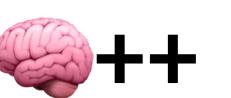
4 HOURS LATER



Carga Cognitiva Extrínseca



¿Dónde puedo encontrar la documentación completa sobre este sistema?



¿Hay algún estándar o plantilla que deba seguir para mi aplicación?



¿Hay algún repositorio específico o rama en la que deba trabajar?



¿Existe una suite de pruebas automatizadas o algún framework de pruebas que deba usar?



¿Cuál es el proceso para desplegar cambios en producción?

Carga Cognitiva

Cómo defino las clases en Python.



Carga Intrínseca

Resolver problemas de configuración que no están directamente relacionados con la tarea principal de consumir mensajes.



Carga Extrínseca

Conceptualizar una arquitectura que sea escalable y pueda atender futuras expansiones o cambios en el proyecto.



Carga Germana

Al **minimizar la Carga Intrínseca** y **eliminar la Extrínseca**,
podemos concentrarnos en la Carga Germana.

Reduce el burn out

Al **minimizar la Carga Intrínseca** y **eliminar la Extrínseca**,
podemos concentrarnos en la Carga Germana.

Incrementa la motivación

Al **minimizar la Carga Intrínseca** y **eliminar la Extrínseca**,
podemos concentrarnos en la Carga Germana.

Fomenta el flujo de desarrollo

Al **minimizar la Carga Intrínseca** y **eliminar la Extrínseca**,
podemos concentrarnos en la Carga Germana.

Trae valor al negocio 

“El propósito de un Platform Team es permitir a los equipos de desarrollo entregar el trabajo con **autonomía**.

[Éste] proporciona **servicios internos** para **reducir la carga cognitiva** que se requeriría de los equipos para desarrollar estos servicios.”

- Team Topologies, Matthew Skelton and Manuel Pais

Componentes de Platform Engineering

**Configuración
de Aplicación**

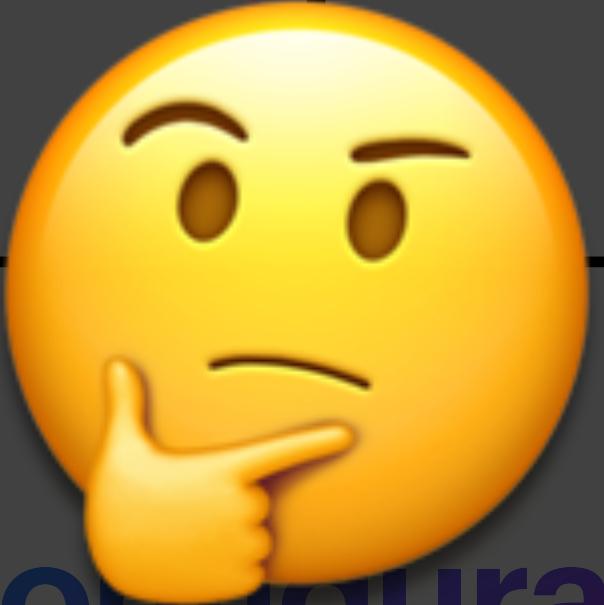
**Orquestación
de
Infraestructura**

**Gestión de
Ambientes de
Desarrollo**

**Gestión de
Despliegue**

**Control de
Acceso Basado
en Roles**

Componentes de Platform Engineering



Configuración
de Aplicación

InfraOps?

Orquestación
de
Infraestructura
Ambientes de
Desarrollo

DevOps?

Control de
Acceso Basado
en Roles

SysAdmin?

Gestión de
Despliegues

**“Platform Engineering no es más que
DevOps con una mentalidad de
producto”**

- Luca Galante

Colaboración Vs. Consumir servicios

“Certo grado de colaboración entre los equipos es esperada, pero a menudo ésta **no escala a toda la organización.**

Consumir cosas-como-servicio suele ser **más eficaz** a medida que el número crece.”

- Team Topologies, Matthew Skelton and Manuel Pais

Internal Development Platforms (IDP)

The screenshot shows the Spotify Internal Development Platform (IDP) homepage. At the top, there is a green header bar with the text "Buongiorno, Kat" and a search bar containing the query "Answer me: Does 'SPTDebugLog' emit logs on iOS test runs?". To the right of the search bar are time zone indicators for NYC (08:30 AM), UTC (12:30 PM), LON (12:30 PM), and STO (08:30 AM). On the far left is a vertical navigation sidebar with various icons. The main content area includes sections for "Favorites" (with links to stack_overflow.questions - Metrics, tpm-contacts.BasicRnDEmployeeData.bq - Overview, Explore the Spotify ecosystem - Platforms, and sciencebox-cloud-notebook-creator - Code Coverage), "Recently Visited" (with links to stack_overflow.questions, sciencebox-cloud-notebook-creator, tpm-contacts.BasicRnDEmployeeData.bq, backstage-data.DocPageProfiles, backstage-frontend, backstage-backend, and events.SearchAPIExposureEvent.gcs), and three call-to-action buttons: "Explore the Ecosystem" (Discover data, apps, platforms and more), "Manage & Maintain" (Take care of the things you own), and "Docs" (Find out how Spotify Tech works). Below these are sections for "News and Updates" (with links to Listen Up! Spotify Has Exciting Plans For An End-To-End Mu..., Today we announce that Playback together with SDK are pr..., and Improving discovery of internal platforms through Backstage...) and "Cycle Time" (a line chart showing cycle time in days over time, with values fluctuating between 2.0 and 3.0).

Service Catalog

The screenshot shows a service catalog interface with a dark sidebar on the left containing various icons. The main header is "Service Catalog" with the subtitle "Keep track of your software". The top right shows time zones for NYC (2:00 PM), UTC (18:00), LON (7:00 PM), and STO (8:00 PM). The navigation bar includes tabs for SERVICES (which is active), WEBSITES, LIBRARIES, DATA ENDPOINTS, APP FEATURES, and DASHBOARDS. Below the navigation is a section titled "Services" with a star icon. On the right are "CREATE SERVICE" and "SUPPORT" buttons. A sidebar on the left lists "PERSONAL" (Owned: 32, Starred: 2) and "SPOTIFY" (All Services: 824). A "Refine Results" section with a "CLEAR" button and dropdowns for "Area" (Python, Go, Java, Data, Websites) follows. The main content area is titled "Owned (32)" and shows a table of services. The table columns are NAME, SYSTEM, OWNER, LIFECYCLE, STATUS, DESCRIPTION, TAG, and ACTIONS. The table lists several services like podcast-api, artist-lookup, searcher, playback-order, shuffle-api, queue-proxy, and playback-order, each with a status indicator (green for up and running), a description, a tag (e.g., Websites, Python, Data, Java), and action buttons (edit, support, star). A large callout bubble in the bottom right corner contains the text: "Gestiona todos tus servicios y componentes de software, en un mismo lugar." (Manage all your software services and components in one place).

NAME	SYSTEM	OWNER	LIFECYCLE	STATUS	DESCRIPTION	TAG	ACTIONS
podcast-api	podcast	Tools	Production	Up and running	Graphql backend for Playlist	Websites	
artist-lookup	artist	Tools	Production	Up and running	Graphql backend for Playlist	Websites	
searcher	search	Tools	Production	Up and running	Graphql backend for Playlist	Python	
playback-order	playback	Tools	Production	Up and running	Graphql backend for Playlist	Data	
shuffle-api	playback	Tools	Production	Up and running	Graphql backend for Playlist	Data	
queue-proxy	playback	Tools	Production	Up and running	Graphql backend for Playlist	Java	
playback-order	playback	Tools	Production	Up and running	Graphql backend for Playlist	Websites	
shuffle-api	playback	Tools	Production	Up and running	Graphql backend for Playlist	Java	
queue-proxy	playback	Tools	Production	Up and running	Graphql backend for Playlist	Websites	

Gestiona todos tus servicios y componentes de software, en un mismo lugar.

Service Catalog

The screenshot shows the 'Service Catalog' interface for the 'playlist-proxy' service. The top navigation bar includes 'Service Catalog > Services > playlist-proxy'. On the right, there are status indicators for 'Owner tools' (Tier 1), 'Service tier' (Tier 1), and 'Lifecycle' (production). The main menu has tabs for 'OVERVIEW' (selected), 'CI/CD', 'TESTS', 'API', 'MONITORING', and 'QUALITY'. A sidebar on the left contains icons for various services and tools.

OVERVIEW

playlist-proxy

Owner tools **Service tier** Tier 1 **Lifecycle** production

OVERVIEW **CI/CD** **TESTS** **API** **MONITORING** **QUALITY**

Overview ★

Graphql backend for Playlist

README

[Getting started](#) [About](#)

To run playlist-proxy, you will need:

- git
- nodeJS
- yarn

After cloning this repo, open a terminal window and start the web app using the following commands from the project root:

```
yarn start  
yarn install
```

[Check out our GitHub →](#)

Pull requests

User	Repository	PR #	Status
	cuepoints/cuepoints	#2809493	Passed
	override/on-demand-trial	#master	Failed
	app-integrations/external	#4238402	Aborted

[View all CI/CD →](#)

Information

[General](#) [Configuration](#)

Owner [#2809493](#)

Support [#playlist](#)

Links [GHE](#)
[Create a plugin](#)
[Plugin Gallery](#)
[backstage.io](#)

Activity

News: Stackoverflow now live! 2019-12-27
After a successful pilot we have decided to invest in...

Product Update: You can now... 2019-12-27

Monitoring

A line chart showing monitoring data over time, with values ranging from 750 to 1250.

- CI/CD
- Tests
- API
- Monitoring
- Quality

Software Templates

The screenshot shows the 'Create a new component' page in Scaffolder. On the left is a vertical sidebar with icons for file operations like search, home, and create. The main area has a teal header with the title 'Create a new component *α*' and a sub-instruction 'Create new software components using standard templates'. Below this is a section titled 'Available templates' with a note: 'NOTE! This feature is WIP. You can follow progress [here](#)'. Two template cards are displayed: 'website React SSR Template' (recommended, React) and 'service Spring Boot Service' (recommended, Java). Both cards have a 'CHOOSE' button at the bottom. In the bottom right corner, there is a callout box with the text: 'Crea nuevos componentes de software en unos pocos pasos, con sus estándares incorporados (Scaffolder)'.

Create a new component *α*

Create new software components using standard templates

Available templates

NOTE! This feature is WIP. You can follow progress [here](#).

website
React SSR Template

Recommended React

Next.js application skeleton for creating isomorphic web applications.

service
Spring Boot Service

Recommended Java

Standard Spring Boot (Java) microservice with recommended configuration.

CHOOSE

CHOOSE

REGISTER EXISTING COMPONENT

SUPPORT

Crea nuevos componentes de software en unos pocos pasos, con sus estándares incorporados (Scaffolder).

The screenshot shows a software application interface for creating a new component. The title bar is teal with the text "Create a new component α " and a subtitle "Create new software components using standard templates". A sidebar on the left contains various icons: a folder, a search magnifying glass, a house, a compass, a list, a plus sign, a gear, a checkmark, a magnifying glass, and a gear with a dot. The main content area is titled "React SSR Template". It displays a step-by-step process:

- 1 Fill in template parameters**
 - Name*: MyNewComponent
Unique name of the component
 - Description*: A great new component
Description of the component
- 2 Choose owner and repo**

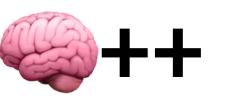
At the bottom of the form are "BACK" and "NEXT STEP" buttons. The "NEXT STEP" button is highlighted with a blue background.

Software Templates

“A los desarrolladores de software les encanta construir plataformas y, sin [...] Product Management, crearán una plataforma más grande de lo necesario.”

- Allan Kelly, Agile

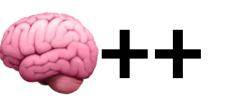
Carga Cognitiva



¿Dónde puedo encontrar la documentación completa sobre este sistema?



¿Hay algún estándar o plantilla que deba seguir para mi aplicación?



¿Hay algún repositorio específico o rama en la que deba trabajar?



¿Existe una suite de pruebas automatizadas o algún framework de pruebas que deba usar?



¿Cuál es el proceso para desplegar cambios en producción?

Adopción

Cognitive load assessment

¿Cómo es la experiencia de construir sus servicios?

*

Cosas a considerar: ¿es la construcción una tarea clara y repetible? ¿Es lo suficientemente rápida? ¿Qué sucede cuando las construcciones fallan? ¿Son fáciles de diagnosticar los fallos?

¿Cómo es la experiencia de probar sus servicios?

*

Cosas a considerar: ¿es la prueba una tarea clara y repetible? ¿Es lo suficientemente rápida? ¿Qué sucede cuando las pruebas fallan? ¿Son fáciles de diagnosticar los fallos? ¿Los entornos de prueba son adecuados? ¿Son fáciles de acceder/iniciar/limpiar/introducir datos de prueba en los entornos de prueba?

Dependencias de los equipos

Team name/focus	Depends on Team	Type (blocking/slowing/ok)	Short description of dependency (artifacts, approvals, other)

Team API

Fecha:

Nombre y enfoque del equipo:

Tipo de equipo:

¿Parte de una Plataforma? (s/n) Detalles:

¿Ofrecemos un servicio a otros equipos? (s/n) Detalles:

¿Qué tipo de expectativas de nivel de servicio tienen otros equipos de nosotros?

Software gestionado y evolucionado por este equipo:

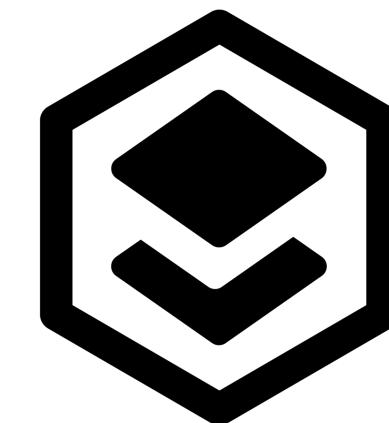
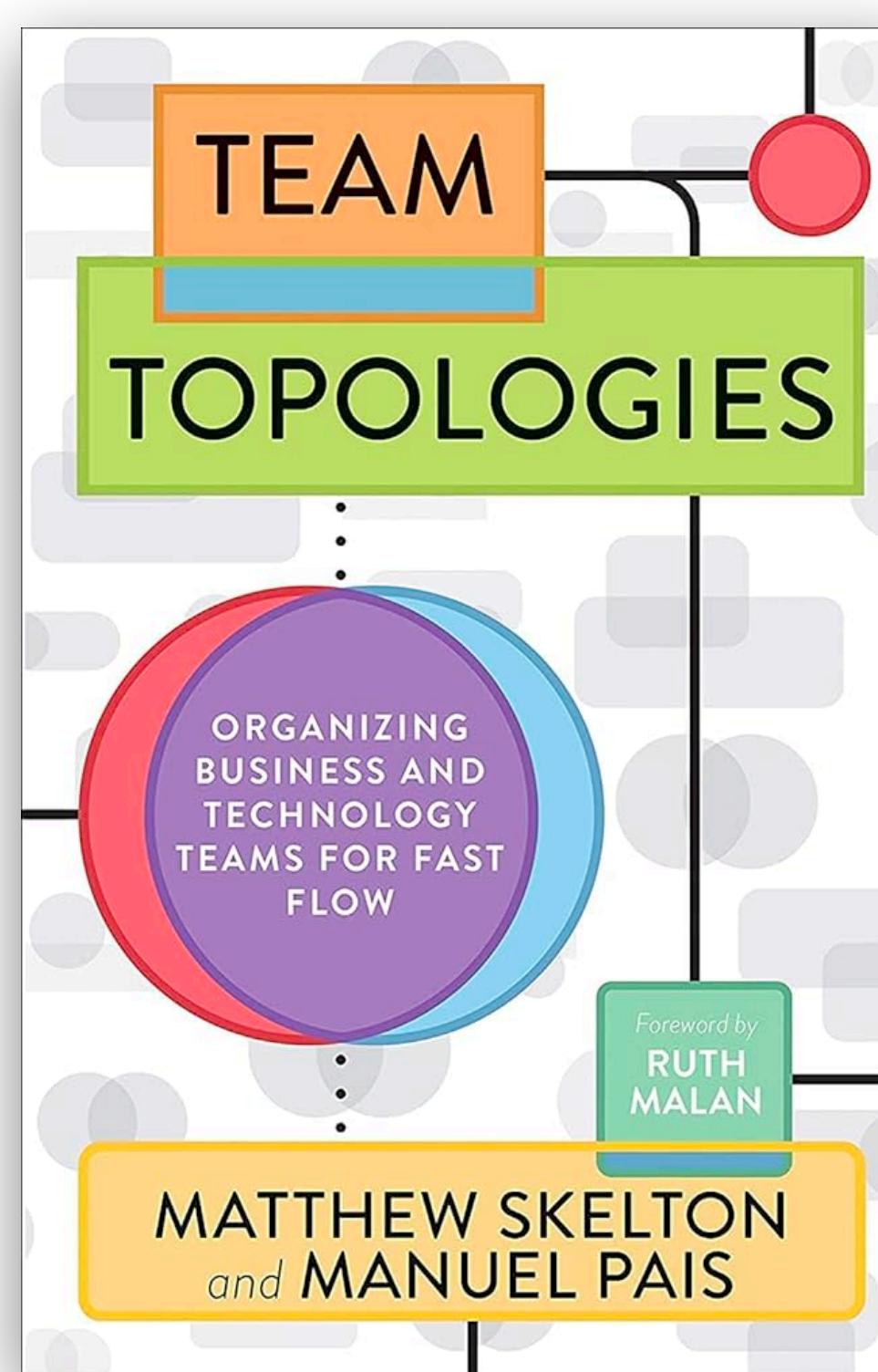
Métodos de versionado:

Términos de búsqueda en la Wiki:

Canales de herramienta de chat: #_____ #_____ #_____

Hora de la reunión diaria de sincronización:

¡Quiero saber más!



Internal
Developer
Platform

PLATFORM
ON23

"Minimizar la carga cognitiva para los demás" es una de las heurísticas más útiles para un buen desarrollo de software.

- Team Topologies, Matthew Skelton and Manuel Pais

¡Gracias!

¿Preguntas?



@alebricio

¡Gracias!

¿Preguntas?

Twitter: @alebricio

LinkedIn: /alejandrabricio