



Servicio de desarrollo de aplicaciones



UNIVERSIDAD
DE MURCIA

Área de Tecnologías de la información
(ATICA)

- El Servicio de Desarrollo de Aplicaciones de ATICA es el encargado de:
 - Desarrollar y mantener la mayoría de **aplicaciones web y móviles** de la Universidad de Murcia
- Está compuesto por aproximadamente:
 - 150 personas
 - 8 equipos especializados en un área de la Universidad de Murcia
 - 350 aplicaciones (y en aumento)



- Cada equipo de trabajo tiene conocimiento especializado en una o varias áreas de la Universidad de Murcia:



Alumnado



RRHH



Investigación



Gestión económica



Docencia



Servicios ofertados



Aplicaciones móviles



UNIVERSIDAD
DE MURCIA

Área de tecnologías de la información
(ATICA)

- Existe también un grupo encargado de gestionar y mantener la pila tecnológica de las aplicaciones:
 - **Diseñar e implantar** la metodología de desarrollo a utilizar ([MEDEA](#))
 - Crear y mantener los **Frameworks de desarrollo**
 - **Investigar** tecnologías emergentes para su incorporación futura
 - Proporcionar métodos y herramientas para **gestionar la seguridad y calidad de los desarrollos**
 - Dar solución a problemas complejos o transversales



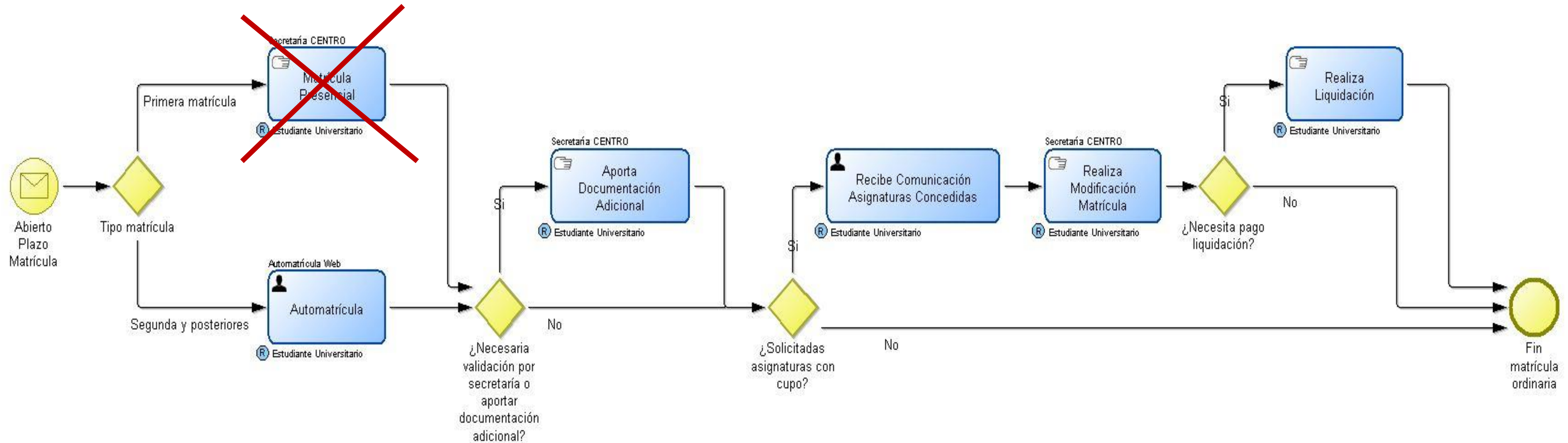
- Además de lo necesario para desarrollar las aplicaciones, el **servicio de infraestructuras** se encarga de:
 - **Proveer los recursos** hardware/software para las aplicaciones:
 - Máquinas físicas
 - Instalación y mantenimiento de servidores
 - Instalación y mantenimiento de bases de datos
 - Configuración de balanceadores, DNS, certificados y políticas de seguridad
 - Configuración de redes y subredes
 - **Monitorizar y mantener** el estado de los servidores
- Sin el servicio de infraestructuras no sería viable desplegar y mantener todas las aplicaciones desarrolladas.



- ¿Cómo de compleja puede ser una aplicación?... Poco... o mucho
 - Aplicaciones que **requieren muchos servicios** de otras aplicaciones
 - El aula virtual para organizar los estudios de un alumno
 - Aplicaciones que requieren **cálculos precisos y complejos**
 - Pago de la nómina de todo el personal de la universidad
 - Aplicaciones que funcionan **basadas en procesos**
 - Necesitan un sistema de gestión de procesos
 - Gran peso de la aplicación se ejecuta en segundo plano
 - Etc.



• ¿Qué ocurre cuando alguien intenta matricularse en la universidad?



→ Servicios externos a los que accede:

- Ministerio de educación
- Dirección General de Policía

→ Implicados en el proceso:

- Alumnos
- Secretarías de los centros
- Plataforma de pago Gurum

- Dado el gran número de proyectos y la complejidad de los mismos se optó por unificar todo en un **único Framework** de desarrollo

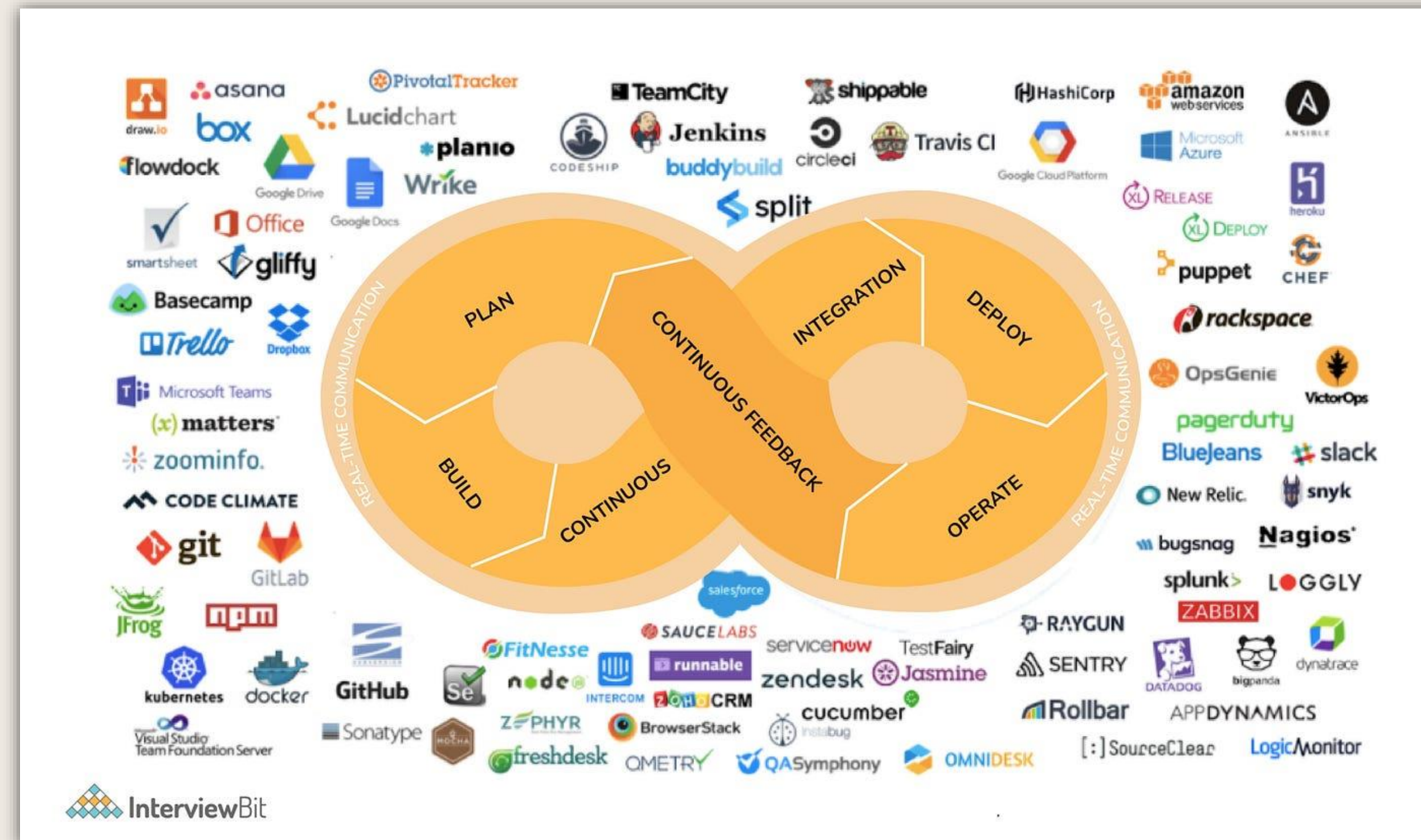


- Pila tecnológica Fundeweb 2.1
 - JSF 2.2
 - JPA 2.0
 - Primefaces 13
 - Seam 2.3
 - HTML5
 - CSS3
 - Servicios web con
 - REST: jaxrw
 - SOAP: jaxws

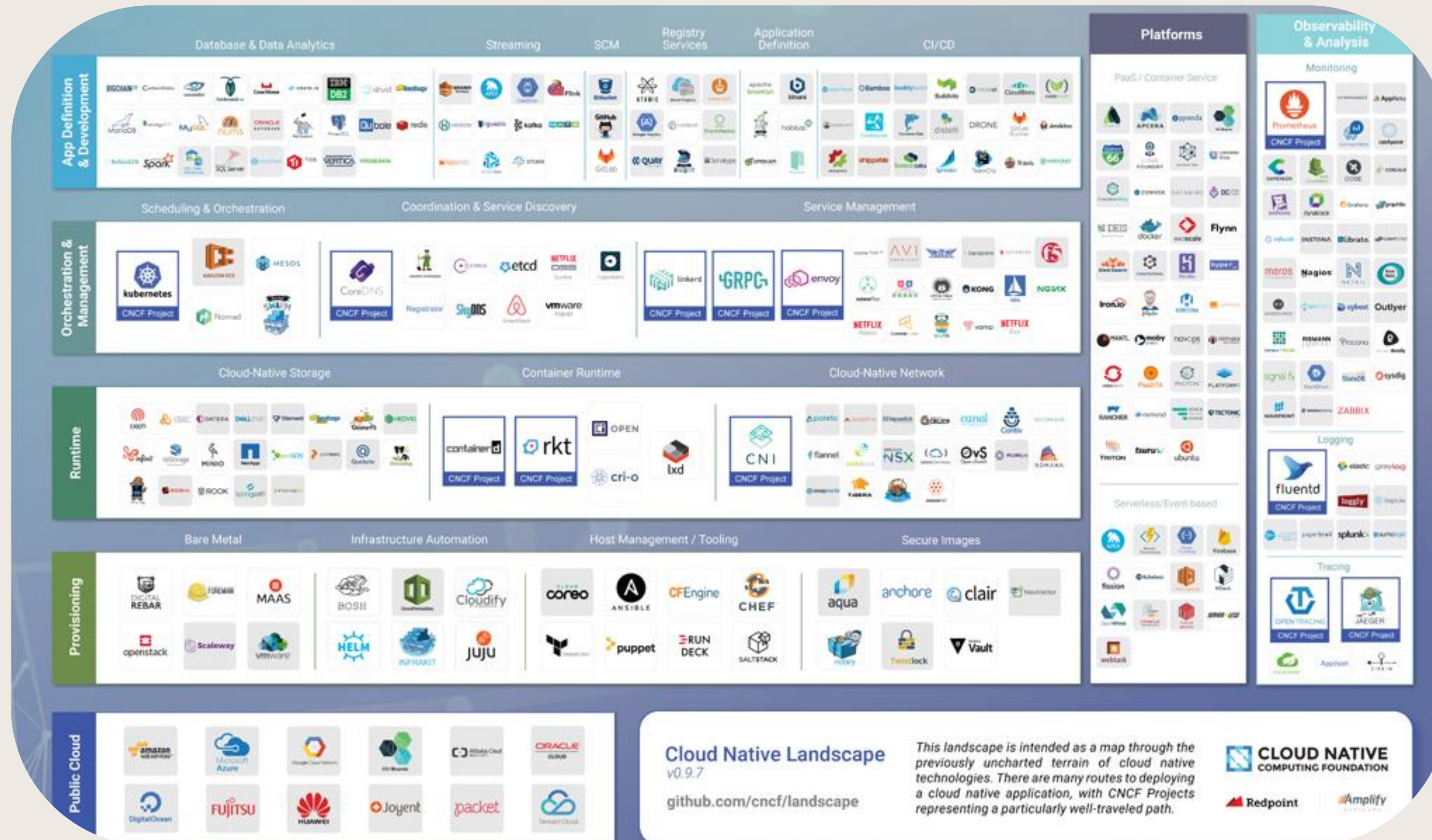


- Pero no sólo el stack tecnológico es suficiente, también necesitamos:

- Sistemas de **gestión del código**
- Sistemas de **control de los despliegues** (CI/CD)
- Mecanismos de gestión de **la calidad del software**
- Mecanismos de aseguramiento de la **seguridad** de los desarrollos



- Framework Fundeweb veámoslo a fondo



- Entorno de desarrollo
 - IDE **Eclipse**:
 - Configuración propia
 - Autoformateado
 - Autooptimización de código
 - Ancho de línea fijo
 - **SpringBoot** Tools
 - Utilidades para proyectos SpringBoot
 - **SonarLint**
 - Análisis de vulnerabilidades, bugs y malas prácticas
 - **Birt**: Generador de reportes (PDF, Word, Excel)
 - **GIT**
 - Servidor Oracle **Weblogic**



- Entorno de desarrollo

- Desarrollo de proyectos:

- Creación a partir de **Arquetipo**

- Todas las librerías ya configuradas
 - El proyecto preparado para arrancar
 - Parte de autenticación y autorización ya implementada
 - Los desarrolladores pueden centrarse directamente en la implementación y no en la configuración

- Configuración mediante **Maven**

- Los proyectos se dividen en 4 módulos siendo los que utilizan los desarrolladores:
 - **Módulo web**: Donde se desarrolla tanto el frontend como el backed
 - **Modulo test**: Donde estarán los test de código

```
1. project
2. |-- pom.xml
3. |-- src
4. |   |-- main
5. |   |-- resources
6. |   |-- archetype-resources
7. |   |-- pom.xml
8. |   |-- src
9. |   |-- main
10. |   |-- java
11. |   |-- App.java
12. |   |-- test
13. |   |-- java
14. |   |-- AppTest.java
15. |   |-- META-INF
16. |   |-- maven
17. |   |-- archetype-metadata.xml
18. |-- test
19. |-- resources
20. |-- projects
21. |-- it-basic
22. |   |-- archetype.properties
23. |   |-- goal.txt
```



- Entorno de desarrollo

- Elección tecnológica

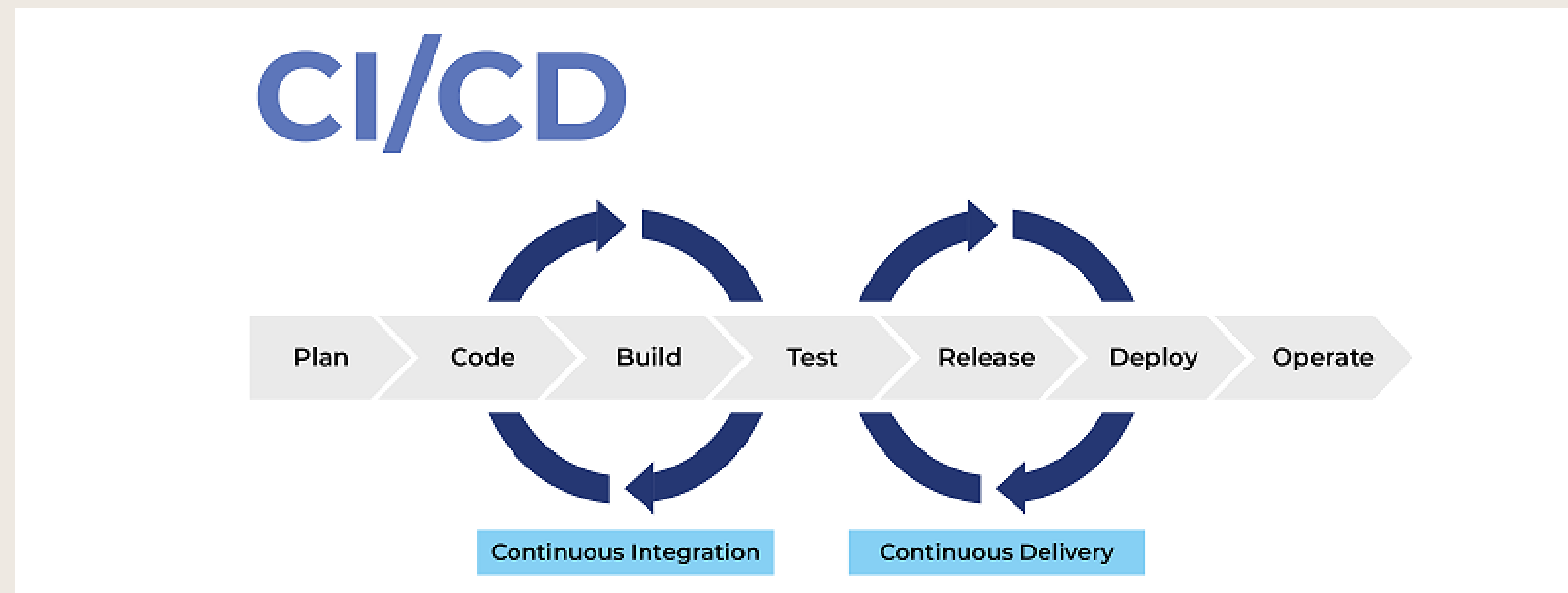
- **JavaJEE**: JSF, JPA, EJB y Primefaces

- ¿Por qué?

- JSF es un **framework robusto** y altamente transaccional
 - El servidor renderiza el HTML que se devuelve por lo que todo el control está en el backend
 - La gestión de la seguridad es más sencilla
 - Permite mantener una **sesión de usuario en memoria** lo que reduce la carga del sistema de autenticación y autorización
 - **Primefaces** como librería de componentes JSF:
 - Framework de componentes JSF altamente extendido
 - Tiene componentes para otros tipos de framework (React, Vue, Angular)



- CI / CD: Continuous integration / Continuous deployment
 - **No podemos subir el código directamente** desde nuestra máquina a los servidores
 - El código debe ser probado y testeado previamente
 - Debemos evitar hacer manualmente las tareas de despliegue:
 - Son repetitivas
 - Serían laboriosas de realizar manualmente
 - Podemos introducir fallos u olvidar paso alargando los despliegues
 - CI / CD nos propone un **sistema automático** para desplegar de manera continua código en los servidores





GitLab


- Gitlab.com como CI / CD base
 - Repositorio GIT de proyectos
 - Provee de un lenguaje para programar flujos de despliegue
 - Al estar desarrollados todos los proyectos con un mismo framework el flujo es el mismo para todos los proyectos
 - Podemos mantener un único pipeline de despliegue, pero desplegar infinitos proyectos, el mantenimiento es mínimo
 - Permite establecer reglas para interaccionar con el código
 - Existen tres ramas GIT base: main, preproducción y producción
 - NO se permite subir código a main ni a preproducción de manera directa sino que tiene que ser aprobado.

Status	Pipeline	Created by	Stages
 Passed 🕒 00:10:47 📅 4 days ago	Merge branch 'jira-PAT-003' into 'desa...' #1074510289  desarrollo  7305cc47 		    



UNIVERSIDAD
DE MURCIA

Área de tecnologías de la información
(ATICA)

- Gitlab.com como CI / CD base
 - Merge Request:
 - No podemos fusionar nuestro código de manera directa con una rama
 - Una Merge Request es una solicitud de fusión
 - Es revisada por quién tenga potestad sobre el proyecto
 - Puede aprobarse, rechazarse o pararse añadiendo comentarios
 - Una vez aceptada el código se fusiona con la rama destino y se lanza el pipeline correspondiente.
- 
- The GitLab logo is located in the top right corner of the slide. It consists of a blue square containing a white stylized 'G' with a red and purple gradient, and the word 'Merge' in white text below it.



New merge request

From `configure-sast` into `jira-FDWL-829` [Change branches](#)

Title (required)

☐ Mark as draft
Drafts cannot be merged until marked ready.

Description

Preview | B I S | | = </> | 🔗 | :≡ := ⌵ ≡ | 📄 | 🛠️ | 🔖

Describe the goal of the changes and what reviewers should be aware of.

Switch to rich text editing 🔧



- Entorno de desarrollo
 - Todos los desarrollos disponen de **4 entornos**:
 - Entorno **Local**: Donde los desarrolladores programan.
 - Entorno **desarrollo**:
 - Dos servidores en cluster con la configuración que deben tener las aplicaciones en los entornos de producción
 - **Recursos limitados**
 - **No se garantiza que todos los servicios estén funcionando**
 - La finalidad es probar que nuestra aplicación arranca correctamente con la configuración de servidor similar a la de producción



- Entorno de desarrollo
 - Todos los desarrollos disponen de 4 entornos:
 - Entorno **preproducción**:
 - Dos servidores en cluster con la configuración que deben tener las aplicaciones en los entornos de producción
 - **Mismos recursos que el entorno de producción**
 - Todos los servicios de este entorno deben estar **funcionando correctamente**
 - En este entorno:
 - Se realizan todas las pruebas de QA de los proyectos
 - Se realizan las “demos” con los clientes finales
 - Entorno de **producción**:
 - Entorno real donde se accede a las aplicaciones



- Hasta ahora hemos provisto de:
 - Un **Framework** de desarrollo
 - Un sistema de **gestión del código** fuente
 - Un sistema **de validación y despliegue** del código
- ¿Qué nos falta?

- Hasta ahora hemos provisto de:
 - Un Framework de desarrollo
 - Un sistema de gestión del código fuente
 - Un sistema de validación y despliegue del código
- ¿Qué nos falta?
 - La **gestión de la QA** de todo el software generado



- ¿Es tan importante gestionar la QA en un desarrollo?
 - Veamos algunos ejemplos





- Boeing 787

- Un **error en la precisión** del tipo de datos de medición provoca que se tengan que reiniciar todo el sistema informático de los aviones
- Tras 248 días de uso ininterrumpido, el software que gestiona los motores causa un desbordamiento por sobrepasar el rango máximo reservado para el tipo de dato.
- **El sistema de emergencia se activa y pone los motores a potencia mínima para evitar daños**
- El problema es que el sistema de emergencia **se activa aunque el avión esté en vuelo**





- Amazon

- Un **fallo tipográfico** al ejecutar un comando, por parte de un administrador, obligó a reiniciar la nube completa de Amazon
- **El hosting estuvo fuera de servicio 5 horas**
- Se vieron afectadas, entre otros: Microsoft, Apple, Aribnb, Netflix ...



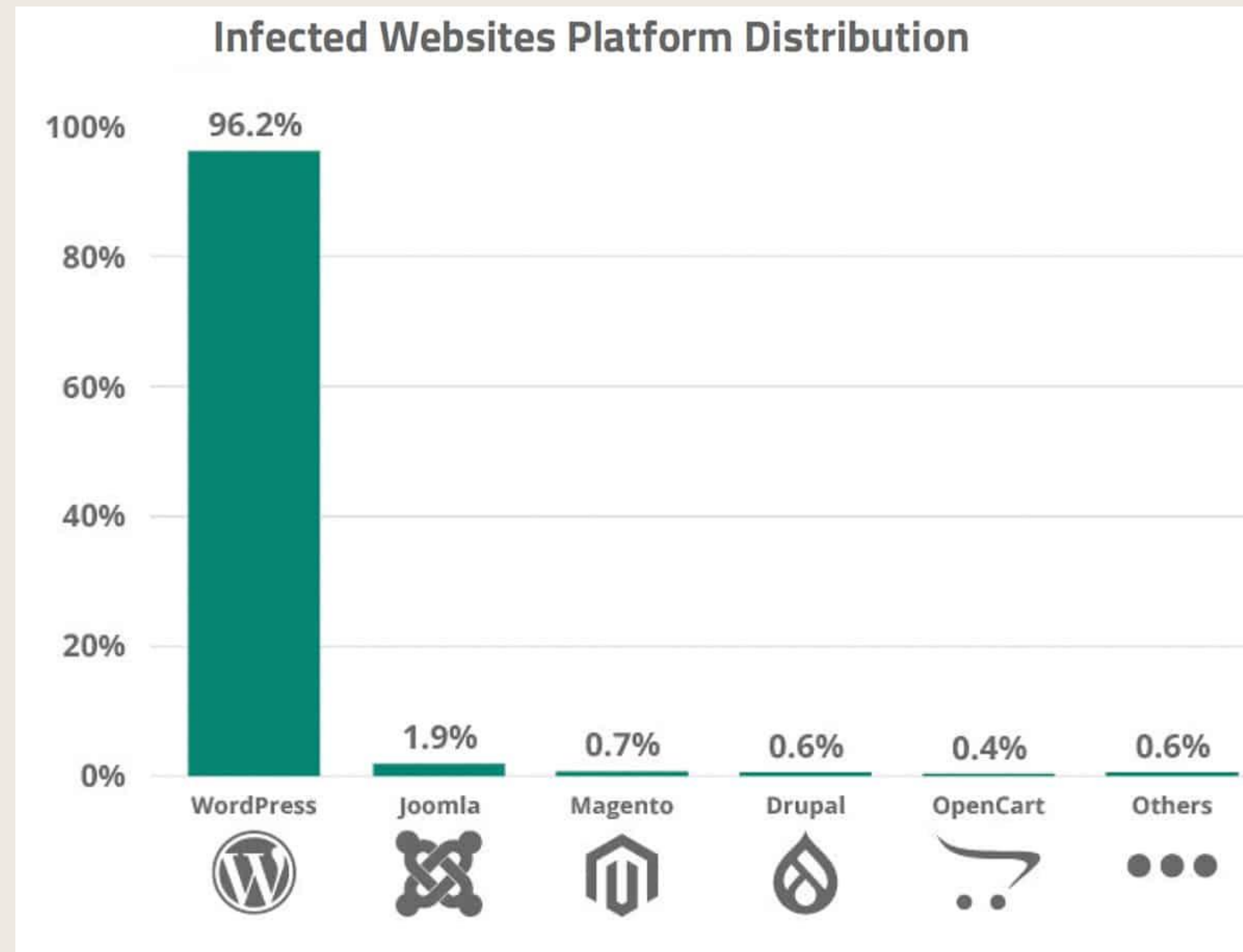


• Toyota

- Los coches Toyota automáticos **aceleraban solos** de manera aleatoria
- En juicio tres técnicos analizaron el código fuente de la centralita y lo calificaron como: "cajón desastre", "cubo de basura" o "plato de espagueti"
- Supuso pérdidas de **más de 4.000 millones de dólares**



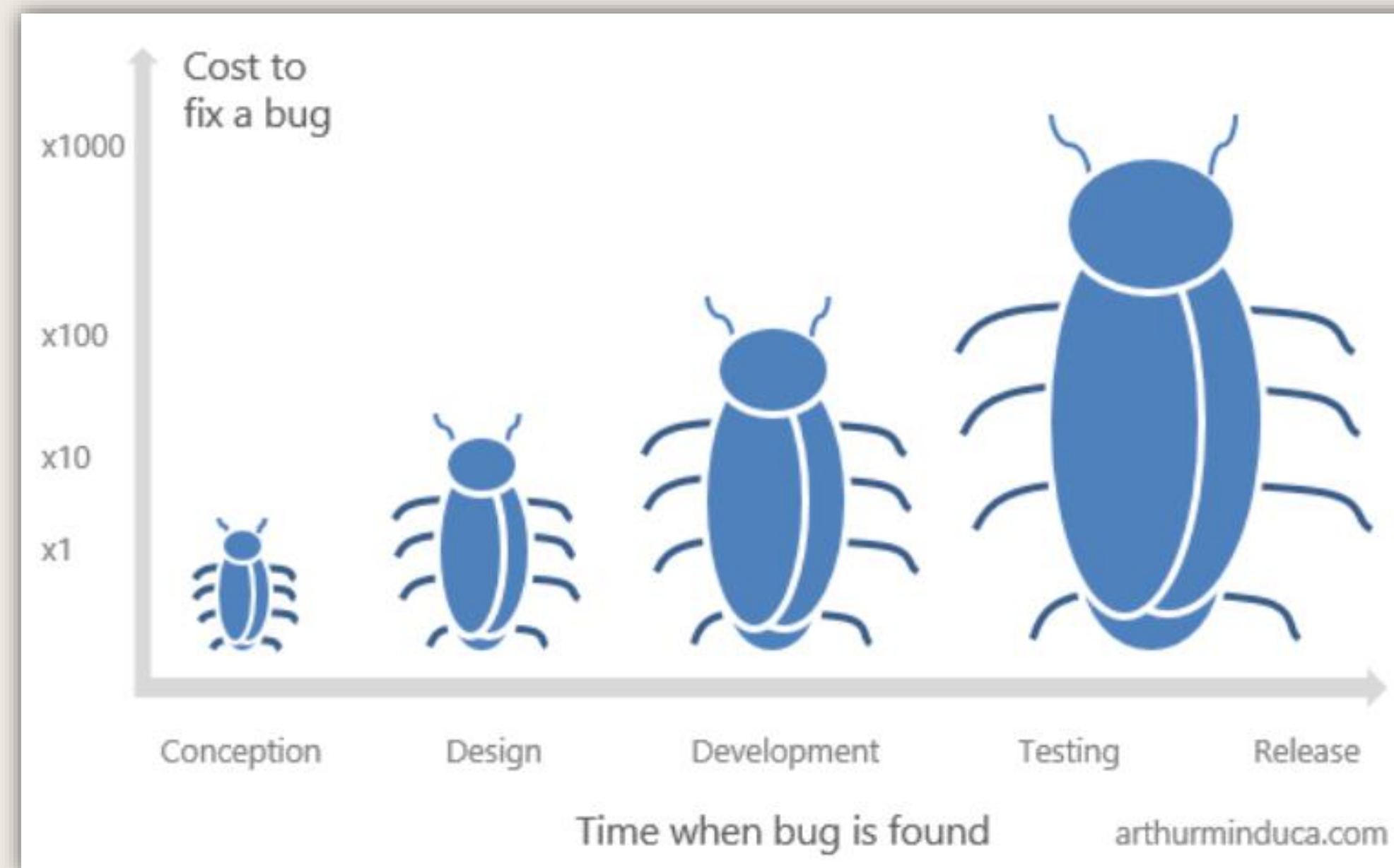
- Si nos centramos en la seguridad
 - ¿Alguien tiene un Wordpress?



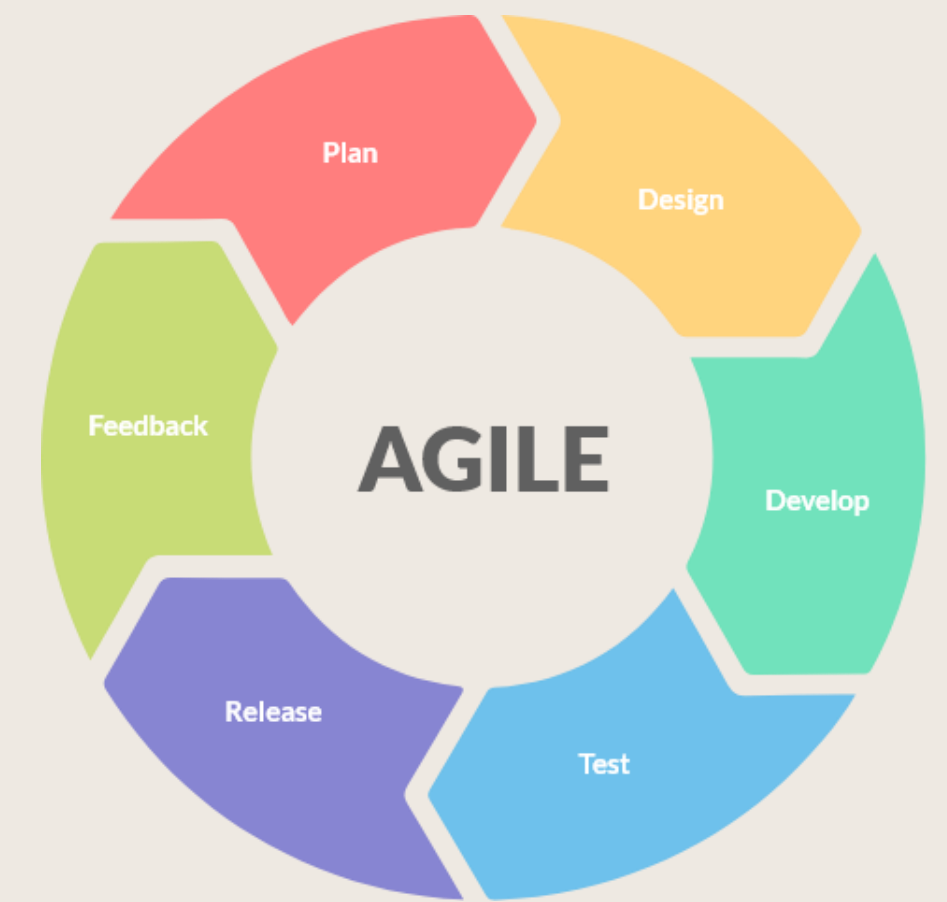
<https://colorlib.com/wp/wordpress-hacking-statistics/>



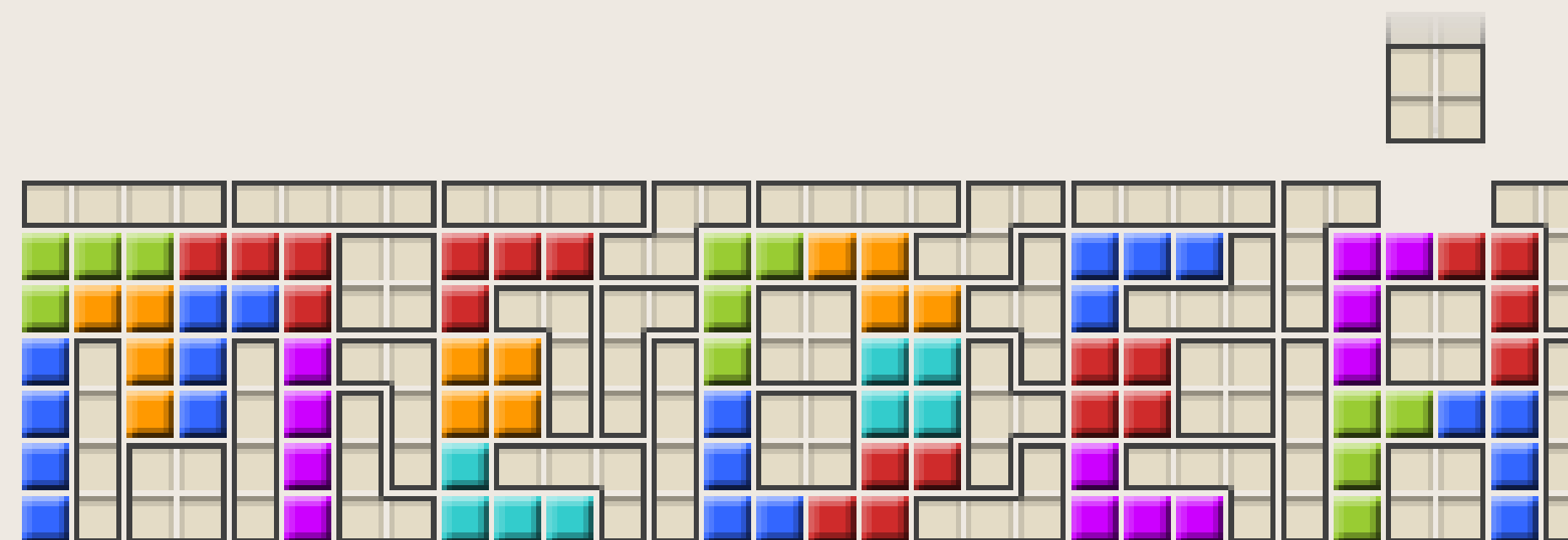
- ¿Qué conclusiones podemos sacar?
 - Es imprescindible **conocer bien los requisitos** del proyecto
 - Los requisitos deben ser:
 - De calidad
 - Exactos
 - Sin ambigüedades
 - En caso contrario cometemos imprecisiones que se arrastrarán hasta el final del proyecto



- Por tanto ¿Qué podemos entender por **gestión de la calidad en el software**?
 - Código correcto
 - Buena documentación
 - Realización de test
 - ... ¿algo más? Sí
 - Definir una **metodología** de desarrollo común
 - Realizar un análisis y **toma de requisitos correcta**
 - Emplear metodologías ágiles (Scrum) que permiten **detectar de manera temprana imprevistos** o situaciones que deben revisarse



- Los desarrollos en ATICA
 - Siguen una **metodología propia basada en metodologías ágiles** existentes pero adaptada al contexto de la Universidad de Murcia
 - Tienen **etapas de QA específicas** para cualquier aplicación o desarrollo
 - Se utiliza **Scrum** con Sprint a 3 semanas de media
 - Cada Sprint acaba con una demo al cliente final



- ¿Qué herramientas utilizamos?
 - Proyectos
 - Eclipse
 - Maven
 - Apache archiva
 - Postman
 - SoapUI
 - JMeter



- ¿Qué herramientas utilizamos?
 - CI / CD
 - Gitlab
 - Gestión de versiones con git
 - Gestión de despliegues con pipelines gitlab



All 1,000+	Pending 88	Running 683	Finished 1,000+	Branches	Tags	Run Pipeline	Clear Runner Caches	CI Lint
Filter pipelines								
Status	Pipeline	Triggerer	Commit	Stages				
running	#146411330		I131649 -> dacc7ea3 Merge branch 'nicolasdular/sto...					
failed	#146410995		I132306 -> 9a5d2aa1 Merge branch '12-10-stable-e...		00:57:08 1 hour ago			
passed	#146410705		I131801 -> 42738af2 Merge branch '210018-remove...		01:26:49 36 minutes ago			
passed	#146410223		Pmaster -> d635c709 Merge branch '22691-externali...		00:00:21 2 hours ago			

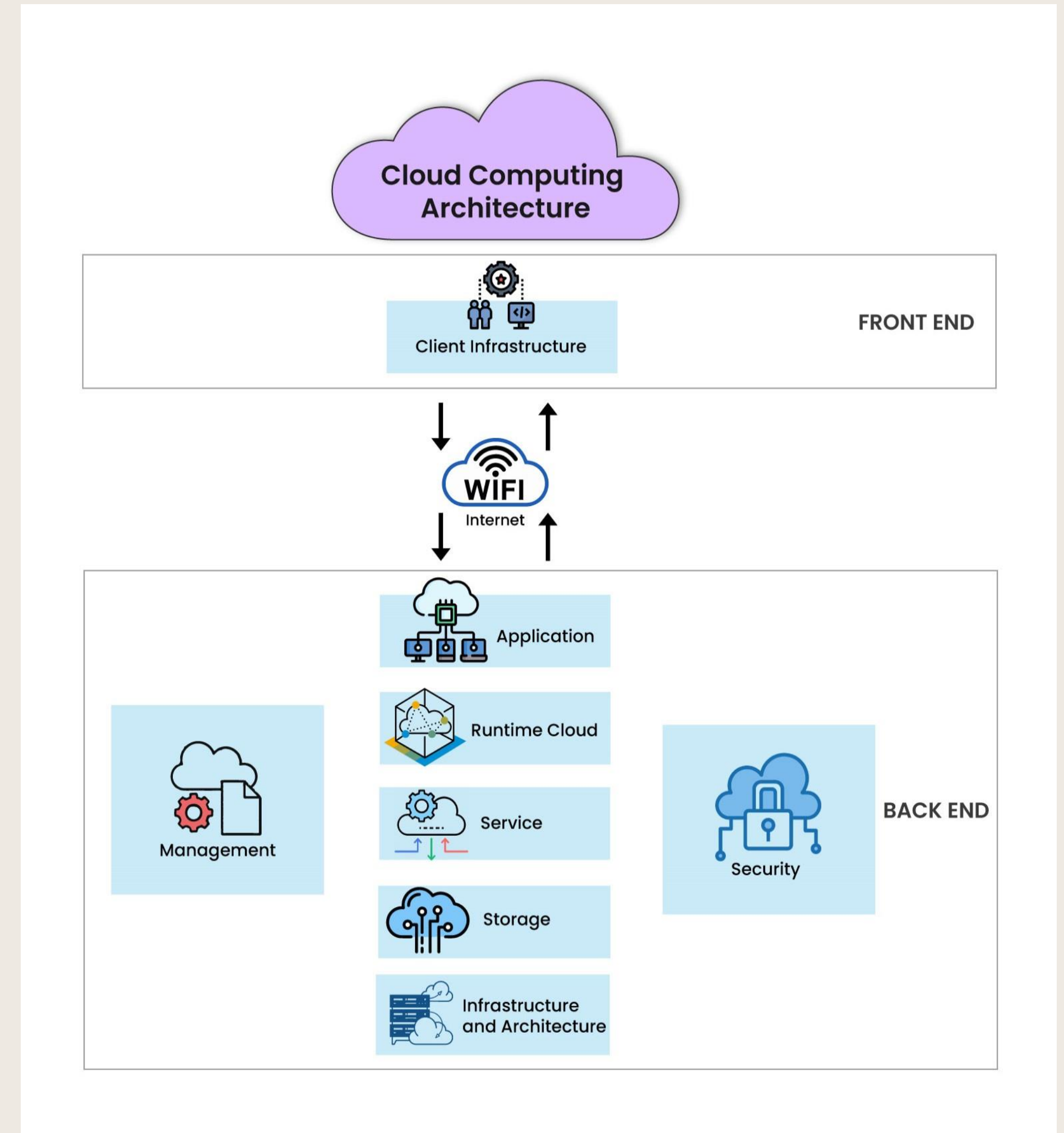
- Pero aún hay más
 - La Universidad de Murcia está encaminándose al **paradigma cloud**
 - Pero ¿qué es el paradigma Cloud o la “nube”?



- Pero aún hay más
 - La Universidad de Murcia está encaminándose al paradigma cloud
 - Pero ¿qué es el paradigma Cloud o “la nube”?
 - Es simplemente el ordenador de otro



- Desarrollo Cloud
 - El principal reto es **el coste**:
 - Consumo de CPU
 - Consumo de memoria RAM
 - Cantidad de tráfico de salida
 - Infraestructura adicional:
 - Dominio
 - Topología de red
 - Bases de datos
 - Gestores de eventos
 - Gestores documentales
 - Gestores de procesos
 - etc



- Desarrollo Cloud

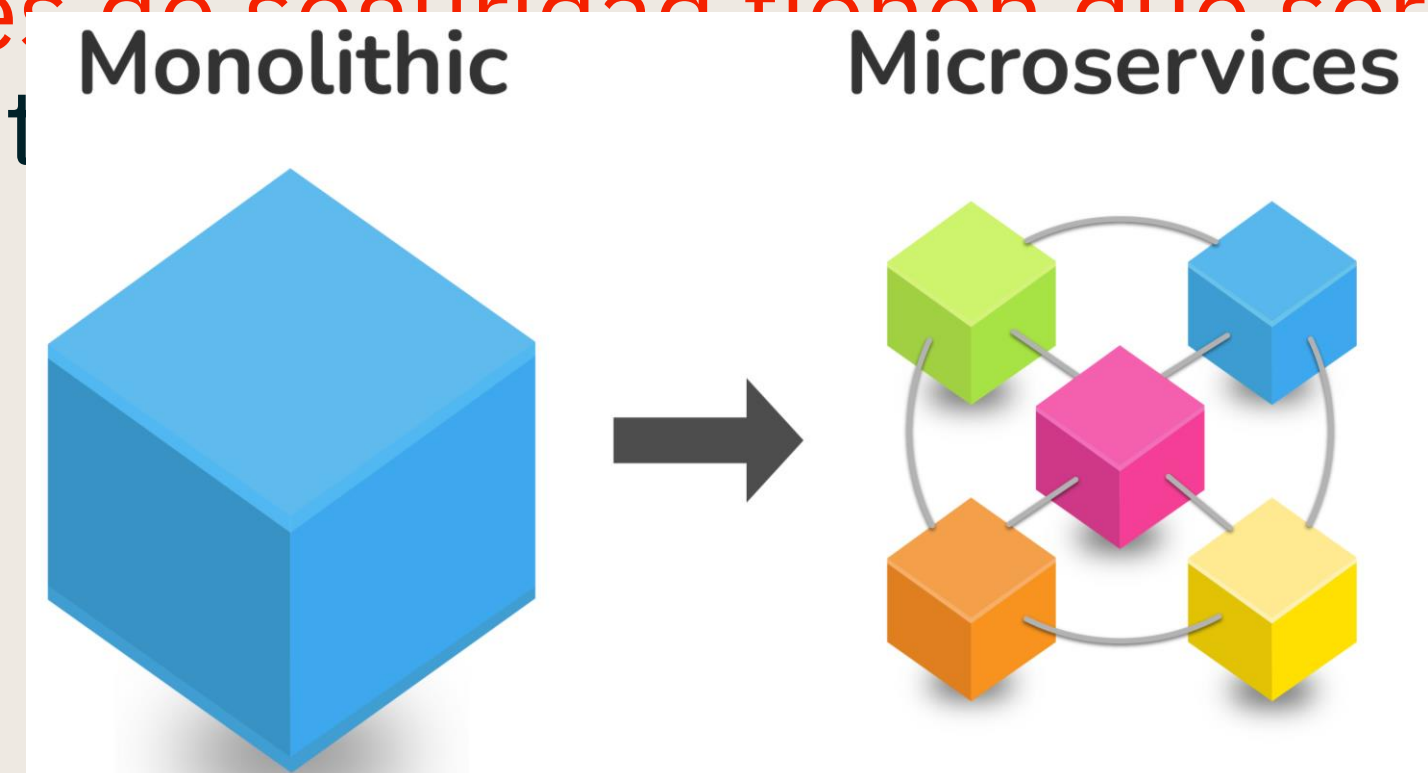
- El Framework Fundeweb es robusto y transaccional pero tiene un **consumo de recursos elevado** por lo que el coste Cloud también lo será
- Solución:
 - Framework **FundewebJS** basado en tecnologías más ligeras
 - **SpringBoot** para el backend
 - **Vue.js** para el frontend
 - Con SpringBoot mantenemos la pila Java pero orientada a servicios REST
 - El intercambio de información será mediante JSON
 - Con Vue.js el peso del renderizado de HTML recae sobre el cliente y no sobre el servidor como con JSF



- Desarrollo Cloud

- Framework FundewebJS

- Los proyectos pasan a estar formados por un **conjunto de microservicios** interconectados
 - Son proyectos pequeños por lo que requieren pocos recursos
 - Fácilmente escalables
 - El coste de renderizado es casi despreciable para los servidores
 - No integrado en el framework
 - Gestión de la sesión de usuario
 - Gestión de la transaccionalidad
 - Se **usarán cookies** para mantener información de estado
 - Las **comprobaciones de seguridad** tienen que ser **más fuertes** ya que la mayoría de servicios son sin estado



- Desarrollo Cloud

- Infraestructura

- Para dar soporte a esta filosofía de despliegue en la nube, necesitamos una **infraestructura cloud propia**
 - Los proyectos ya no es sólo código:
 - Es un empaquetado **Docker** con su propio sistema operativo, configuración y código fuente
 - El sistema de despliegue ya no sólo deja el empaquetado en el servidor:
 - Se usa la filosofía IaC (**Infrastructure as Code**) donde a partir de ficheros yml se deciden los detalles de la infraestructura
 - CPU asignada
 - Memoria RAM dedicada
 - Unidades de disco montadas
 - Configuración de red
 - Etc.



- Desarrollo Cloud
 - Infraestructura
 - Ahora cuando desplegamos un proyecto
 - Se construye una imagen que consiste basada en Docker con el software a construir (desde el sistema operativo hasta la aplicación) y su configuración
 - Se obtiene la descripción del “hardware” de soporte a partir de un repositorio de configuración y ficheros yml
 - Se le dice al gestor de contenedores Kubernetes que despliegue una nueva máquina según las indicaciones de la imagen y la configuración
 - Además, en la configuración se indica
 - Cómo y cuando escalar la imagen
 - Cómo y cuando desescalarla
 - Qué límites de tráfico va a tener
 - Etc.



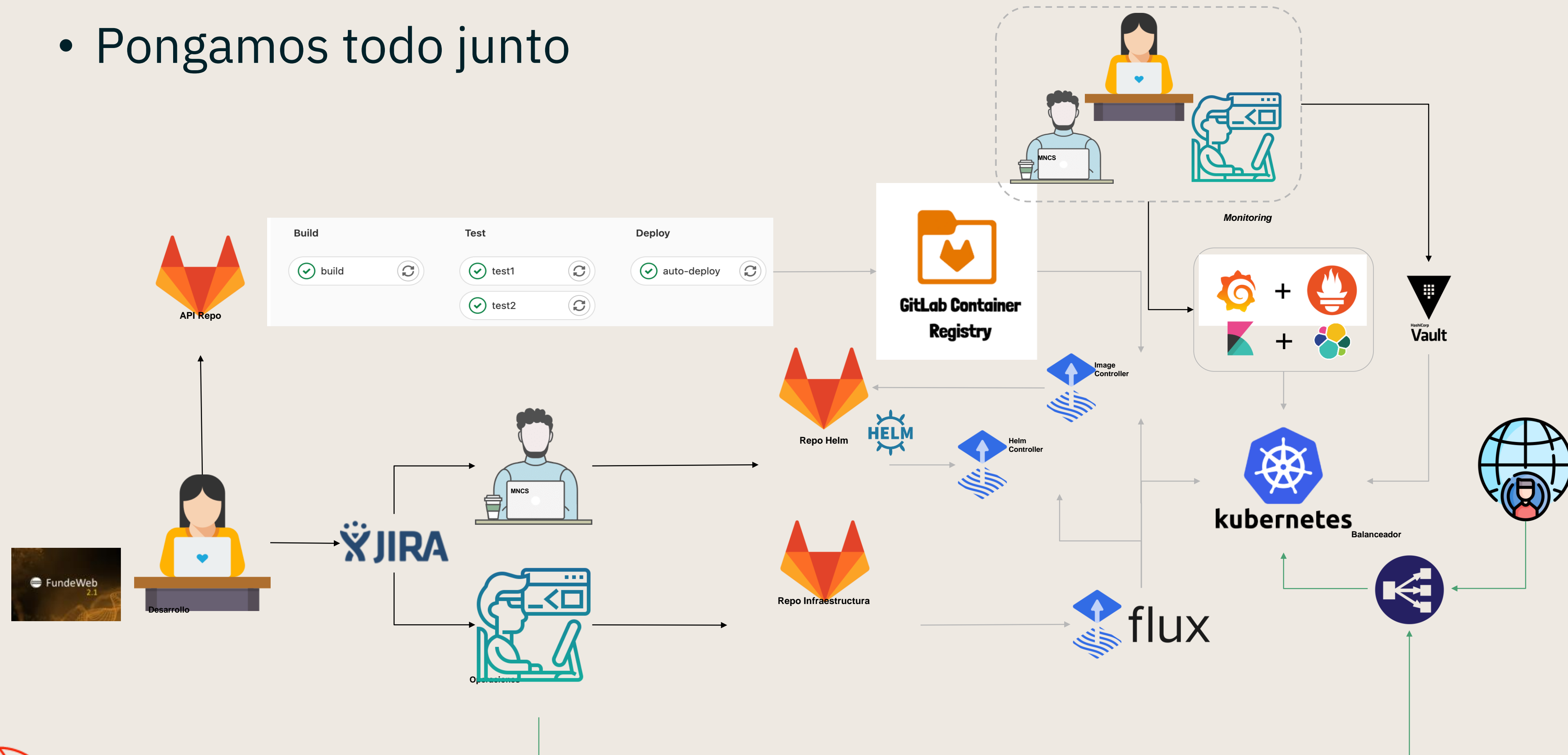
kubernetes



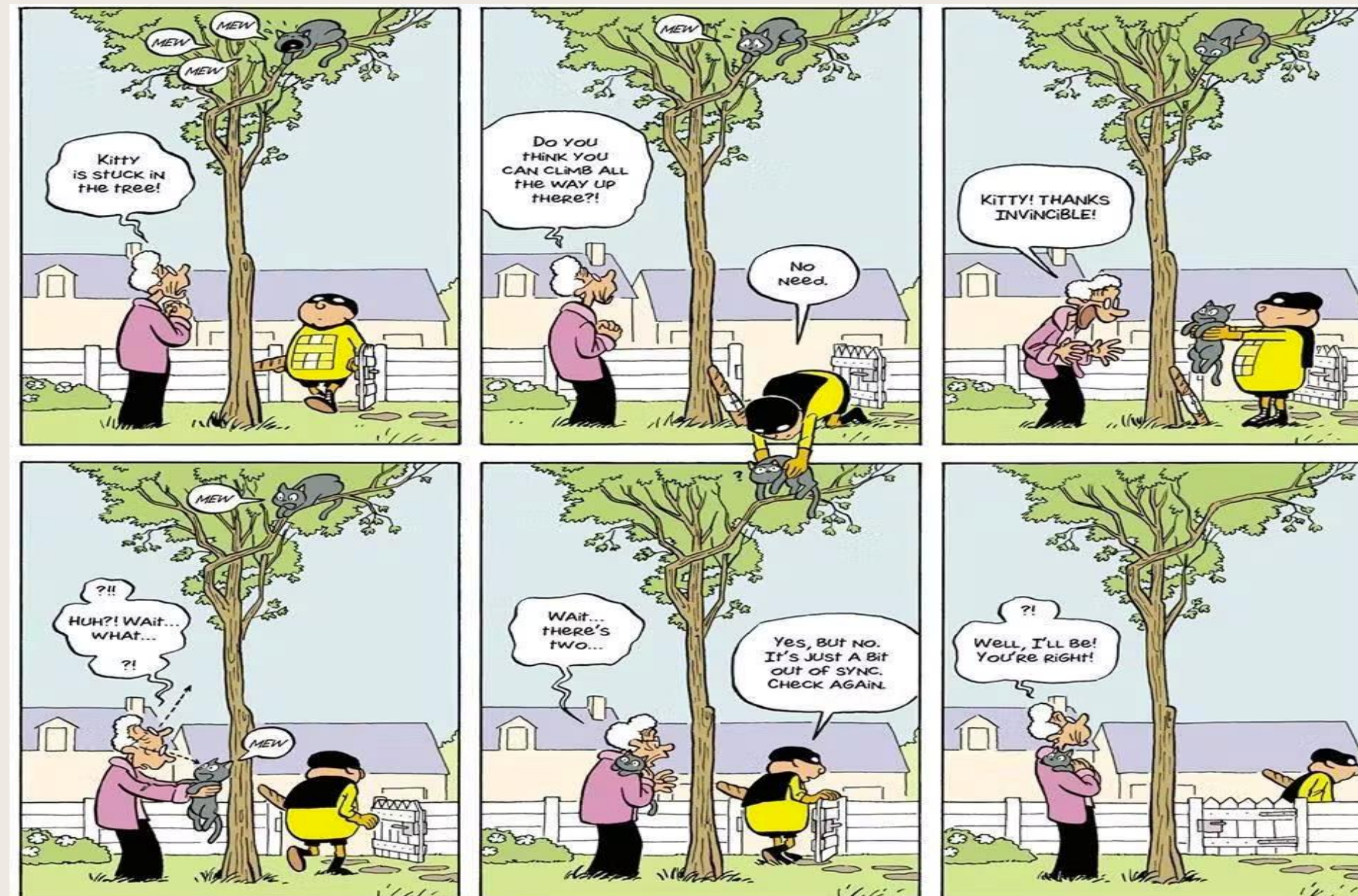
UNIVERSIDAD
DE MURCIA

Área de tecnologías de la información
(ATICA)

- Desarrollo Cloud
 - Pongamos todo junto



- Desarrollo Cloud
 - El objetivo final a alcanzar es la **comunicación vía eventos**
 - No habría llamadas a servicios concretos
 - Existirá un pool de eventos que escucharán todos los servicios
 - El hándicap es conseguir que el sistema soporte **consistencia eventual**



- Desarrollo en ATICA, resumen
 - Equipos de desarrollo especializados en las diferentes áreas de la Universidad de Murcia
 - Dos Frameworks de desarrollo según las necesidades del proyecto
 - En el mundo de la informática no hay una única solución para todo
 - Diferentes entornos para probar las aplicaciones
 - Metodología de desarrollo propia
 - Fuerte gestión de la QA de los desarrollos
 - Avanzando hacia un paradigma cloud

Muchas gracias
¿dudas o preguntas?



