SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Gobierno del Estado de Sonora

SEC
Secretaría de Educación y Cultura

Universidad Tecnológica del Sur de Sonora

Todo Computación

Elaboraron:
Itzel Samantha Palafox Cuen
Reyna Belem Buitimea Galaviz
Carlos Ruben Vazquez Padilla

Materia:
Gestión del Proceso de Desarrollo de Software

Docente:
Mtro.Tadeo Portela

Grupo:
IDGS 10°A

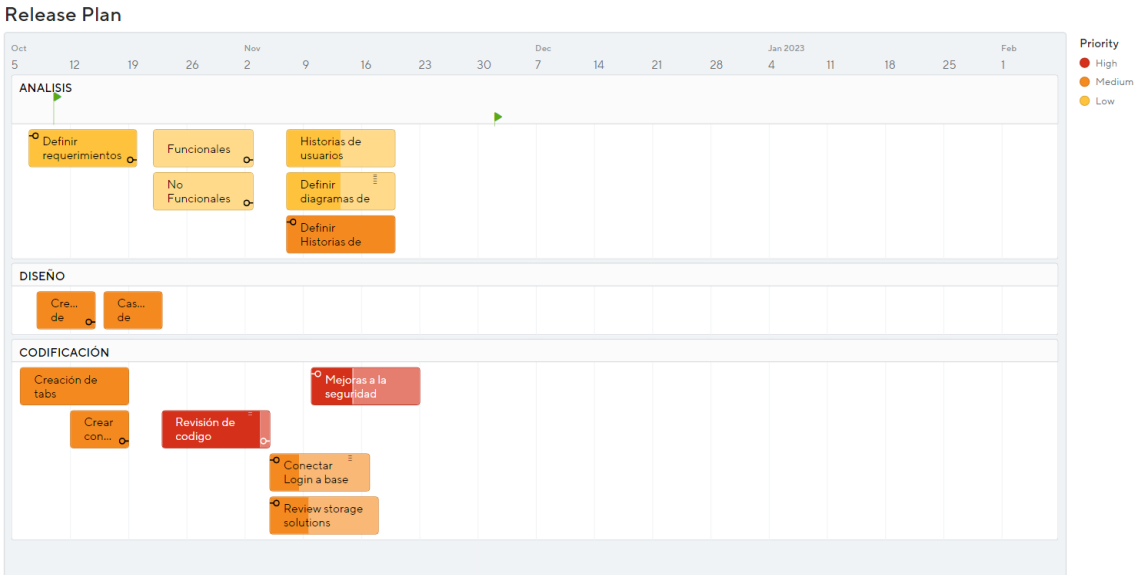
CD. OBREGÓN, SONORA **OCTUBRE 2022**

1 .Genera un documento a partir de un caso de estudio el cual incluya:

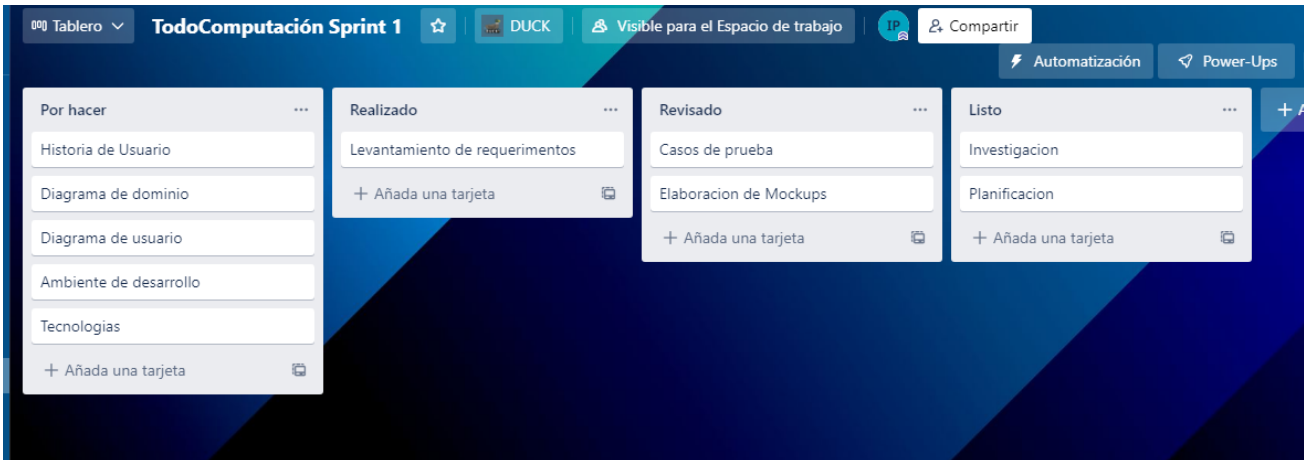
- Pila del producto

Id	Tarea	Descripción	Prioridad	Estado
1	Crear una cuenta de logeo	El usuario podrá crear una cuenta para interactuar con el sitio teniendo más opciones de compra.	Alta	
2	Actualizar datos de la cuenta	El usuario podrá modificar su información.	Alta	
3	Realizar compras	El usuario podrá hacer compras de los productos en el sitio.	Alta	
4	Modificar compras	El usuario podrá agregar o quitar productos de sus compras.	Alta	
5	Detalle de compras	El usuario recibirá un detalle de las compras realizadas.	Media	
6	Agregar productos	El administrador podrá agregar más productos en existencia al sitio.	Alta	

-Plandeliberación



- Plan de Sprint 1



- Plan de pruebas - para una iteración

1. Pruebas estáticas

Durante las pruebas estáticas, los desarrolladores trabajan para evitar posibles problemas que puedan surgir posteriormente. Sin ejecutar el código, realizan revisiones manuales o automatizadas de los documentos de apoyo al software, como las especificaciones de requisitos, en busca de posibles ambigüedades, errores o redundancias. El objetivo es adelantarse a los defectos antes de introducirlos en el sistema de software.

2. Pruebas unitarias

La siguiente fase de las pruebas de software es la prueba unitaria. Durante esta fase, el software se somete a evaluaciones de sus unidades específicas, o de sus funciones y procedimientos, para garantizar que cada una de ellas funciona correctamente por sí misma. Los desarrolladores pueden utilizar las pruebas de caja blanca para evaluar el código y la estructura interna del software, normalmente antes de entregar el software para que lo prueben formalmente los probadores. Las pruebas unitarias pueden tener lugar siempre que un fragmento de código sufra cambios, lo que permite resolver rápidamente los problemas.

3. Pruebas de integración

Las pruebas de integración consisten en probar todas las unidades de un programa como un grupo para encontrar problemas con la forma en que las funciones de software separadas interactúan entre sí. A través de las pruebas de integración, los desarrolladores pueden determinar la eficiencia global de las unidades cuando se ejecutan juntas. Esta fase es importante porque la funcionalidad global del programa depende de que las unidades funcionen simultáneamente como un sistema completo, no como procedimientos aislados.

4. Pruebas de sistemas

En la fase de prueba del sistema, el software se somete a su primera prueba como aplicación completa e integrada para determinar si cumple su propósito. Para ello, los desarrolladores pasan el software a probadores independientes que no han intervenido en su desarrollo para garantizar que los resultados de las pruebas proceden de evaluaciones imparciales. Las pruebas del sistema son vitales porque garantizan que el software cumple los requisitos determinados por el cliente.

5. Prueba de aceptación

Las pruebas de aceptación son la última fase de las pruebas de software. Su objetivo es evaluar la disposición del software para su lanzamiento y uso práctico. Los probadores pueden realizar las pruebas de aceptación junto a personas que

representan al público objetivo del software. El objetivo de las pruebas de aceptación es demostrar si el software satisface las necesidades de los usuarios a los que va dirigido y si los cambios que ha experimentado el software durante su desarrollo son adecuados para su uso. Las personas representativas son cruciales en esta fase porque pueden ofrecer una visión de lo que los clientes pueden querer del software. Una vez que el software supera las pruebas de aceptación, pasa a producción.

- Casos de pruebas - para una iteración

- **Positivos**: Pruebas destinadas a verificar el funcionamiento correcto de la funcionalidad utilizando el formato de entrada correcto:
    - Por ejemplo verificar los formatos de correos electrónicos: (letras permitidas, caracteres

  especiales permitidos, números,)

- **Negativos**: Pruebas destinadas a verificar el funcionamiento correcto de la funcionalidad utilizando

  el formato de entrada incorrecto, esperando como resultado un mensaje de error.

    - Por ejemplo, verificar los formatos de correos electrónicos: (caracteres especiales que no están permitidos) se debe validar que mediante mensajes informativos se tenga conocimiento de la falla.
- **Valor límite:** Pruebas destinadas a verificar el funcionamiento correcto de la funcionalidad utilizando formatos permitidos y no permitidos.
    - Por ejemplo verificar los formatos de correos electrónicos: si se tiene contemplado que en

el campo de correo el número de caracteres no supere los 20 dígitos antes del símbolo @ se debe validar la respuesta de la aplicación, al digitar diferentes valores, incluyendo espacio.

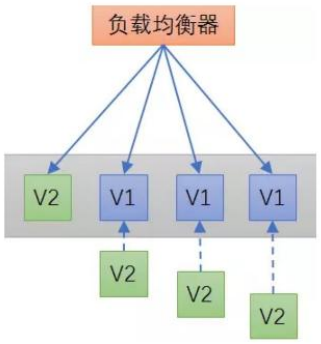
- Estrategia de despliegue - Investigar y definir cuál es la más adecuada para su proyecto.

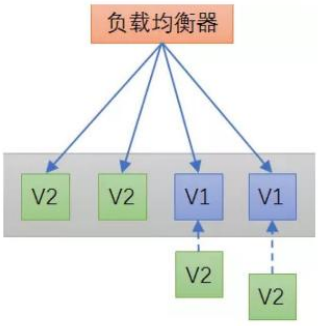
***Actualiza rodante o continua.**

Es una mejora en el modelo de implementación canaria. Se caracteriza por una actualización continua continua de un solo servicio. El primer método de actualización de servicio es la implementación canaria.

Ventajas: pequeño impacto en el usuario, experiencia fluida

Desventajas: el tiempo de lanzamiento / reversión es largo y debe ser compatible con la herramienta de lanzamiento. Por ejemplo: k8s



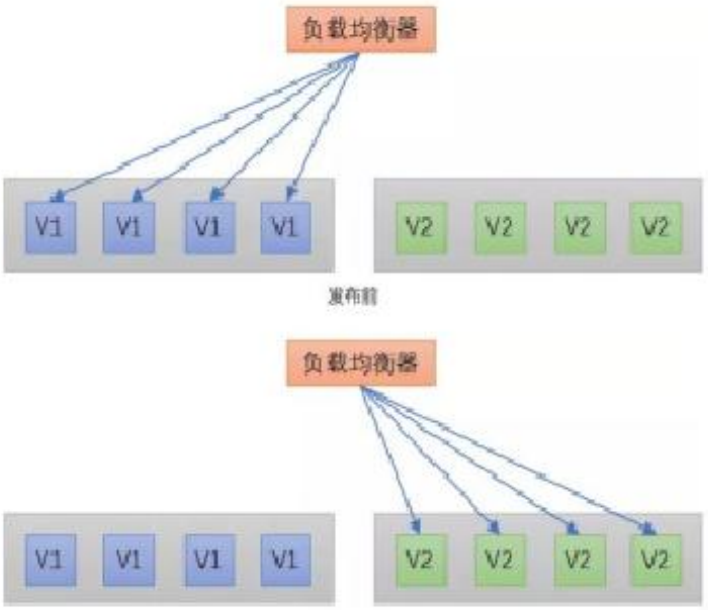


***Lanzamiento azul/verde.**

Es un escenario adecuado para la publicación de grupos de servidores dobles: se caracteriza por ejecutar un grupo de servicios en línea, reconstruir un conjunto de servicios actualizados fuera de línea y finalmente cambiar para completar la operación de publicación a través del tráfico único.

Ventajas: cambio rápido de actualización / velocidad de retroceso

Desventajas: cambio completo, gran influencia del usuario, recursos dobles de la máquina



Dado que es un grupo de servidor dual, los servicios de los dos grupos son diferentes; un grupo es el grupo azul y el otro grupo es el grupo verde, por lo que se llama implementación azul-verde.

***Lanzamiento de prueba A / B**

Un método de lanzamiento que permite que existan múltiples versiones de servicios en línea al mismo tiempo. El propósito es verificar la efectividad de la

nueva versión del servicio y evaluar el éxito de la nueva versión a través del tráfico real de usuarios en línea.

Generalmente se aplica a la publicación del servicio de efectos y productos.

Ventajas: utiliza el tráfico real para verificar, tiene poco impacto en los usuarios y puede ser probado por personas específicas

Desventajas: publicación compleja y plataforma de recopilación de datos

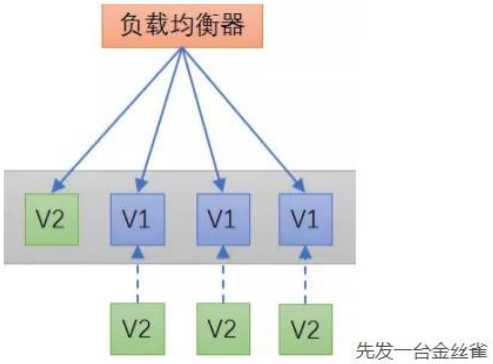
***Lanzamiento Canario**

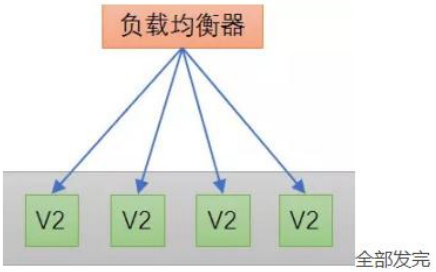
Un método de lanzamiento mejorado basado en el proceso de lanzamiento tradicional. La característica es detener un servicio para actualizar primero y observar durante un tiempo después de conectarse.

Después de ningún problema, actualice todos los servicios restantes a la vez. El primer servicio actualizado es "Canarias".

Ventajas: el proceso de lanzamiento y las anomalías de lanzamiento tienen un pequeño impacto en los usuarios

Desventajas: el servicio seguirá interrumpido y el ciclo de lanzamiento / reversión es muy largo.





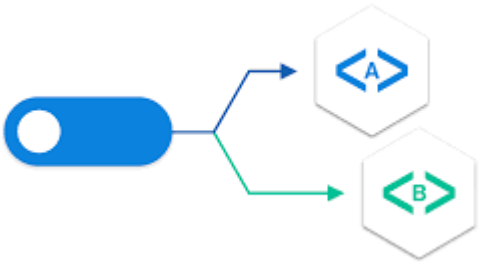
El origen de "Canario" es que antes del absentismo para abrir la mina a la mina, primero se colocará un canario para averiguar si hay gas tóxico, para ver si el canario puede sobrevivir, y el canario lleva su nombre.

***Feature Flags.**

En procesos modernos como en desarrollo ágil es común que se hagan iteraciones para entregar nuevas funcionalidades a la aplicación, muchas veces es posible entregar un sección completa pero otras veces solo se realiza una parte, quedando incompleto, por ejemplo:

Supongamos que tenemos una tarea dedicada a construir la base de una página pero el resto de elementos por su complejidad serán implementadas en otra tarea, dejándola así incompleta para el usuario final. Continuando con las buenas prácticas, nos gustaría agregar el código al branch principal ("continuous integration" y "continuos deployment") ¿pero que vamos hacer con esta página que está incompleta? Aquí es donde entra en juego los feature flags.

Los feature flags también son utilizados en estrategias de A/B testing en el que puedes mostrar cierta funcionalidad(feature/característica) a una parte de la población de usuarios y a otra no. Esto es una estrategia de marketing que permite descubrir que es más atractivo/visitado/utilizado por el usuario.



***Pruebas A/B.**

Las pruebas a/b tienen el objetivo de analizar cuál es el contenido que mejor atiende a nuestro público.

Por ejemplo, si quieres probar la efectividad de un Email Marketing, antes de enviarlo a todos, envíalo a un 5% de tu base de datos y espera.

Después de más o menos 24 horas es posible analizar cuál es la campaña que ha resultado más efectiva para que puedas enviarla a todo el resto de tu audiencia.

Este tipo de acción es muy utilizada en estrategias de marketing, en campañas de landing page y otros.

Podemos utilizar las pruebas a/b también creando distintas versiones de un elemento que aleatoriamente será mostrado a los usuarios para saber cuál rinde mejor.

Prueba A:

En la versión A deciden crear unos diseños en forma de burbujas de conversación en donde se explican los beneficios de reservar boletos de avión y habitaciones de hotel, dentro de una conversación de dos personas con imágenes de stock.

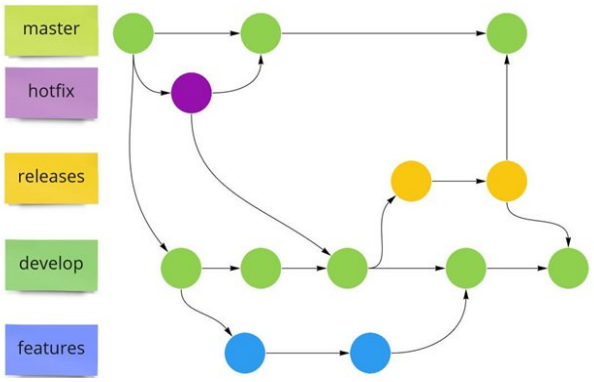
Prueba B:

En la prueba B o versión B, escogen un diseño más "sobrio" sin menú y donde posicionan el título atractivo y preciso: "¡Déjanos ocuparnos de tu viaje!".

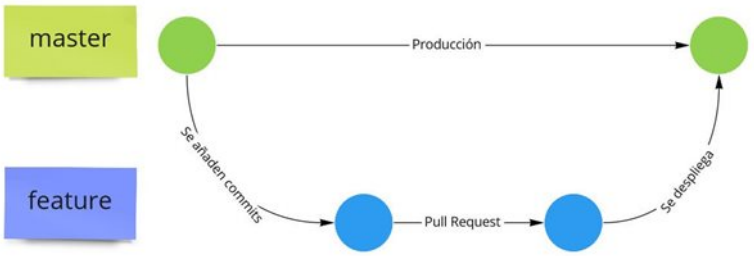
Luego, despliega un texto en donde resaltan la autoridad de la marca diciendo: "Viajes Tu Mundo es una empresa líder en el segmento turístico mexicano, ayudando a miles de personas a encontrar destinos y precios acordes a sus expectativas sin los molestos procesos de reservas que son un dolor de cabeza para muchos".

- Flujo de trabajo para el control de versiones (Git Workflows) - Investigar y definir cuál se adapte mejor a los procesos de Integración Continua.

*Git Flow: Es el flujo de trabajo más popular y extendido. Se basa en dos ramas principales con una vida infinita. Para cada tarea que se le asigna a un desarrollador se crea una rama feature en la cual se llevará a cabo la tarea. Una vez que ha finalizado, realizará un pull request (validación) contra develop para que validen el código.



*GitHub Flow:



La diferencia principal con Git Flow es la desaparición de la rama develop. Se basa en los siguientes principios:

-Todo lo que haya en la rama master debe ser desplegado.

-Para cualquier característica nueva, crearemos una rama de master, usando un nombre descriptivo.

-Debemos hacer commit en esta rama en local y hacer push con el mismo nombre en el server.

-Si necesitamos feedback, utilizaremos las herramientas de mergeo como pull request.

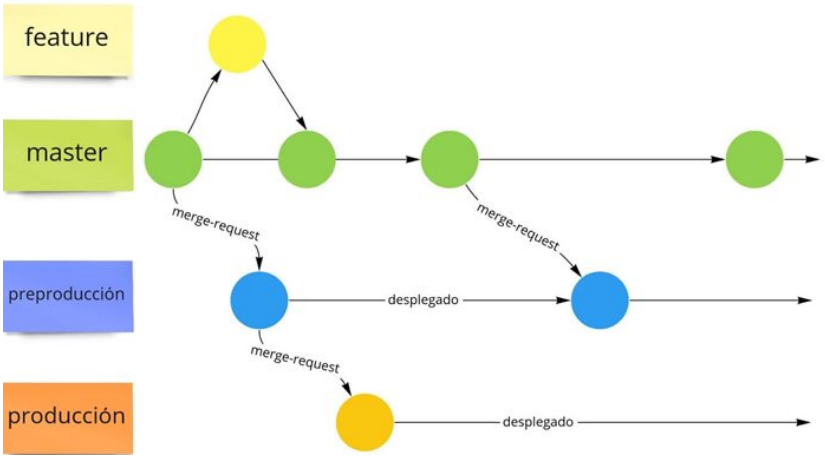
-Una vez revisado el código, podemos mergear contra master.

-Una vez mergeado contra máster, debemos desplegar los cambios.

*GitLab Flow:

GitLab Flow es una alternativa/extensión de GitHub Flow y Git Flow, que nace debido a las carencias que adoptan estos dos flujos. Mientras que una de las consignas de GitHub es que todo lo que haya en master es desplegado, hay ciertos casos en los que no es posible cumplirlo o no se necesita. Por ejemplo: aplicaciones iOS cuando pasan a la App Store Validation o incluso tener ventanas de despliegue por la naturaleza del cliente.

El flujo propone utilizar master, ramas features y ramas de entorno. Una vez que una feature está finalizada hacemos una merge request contra master. Una vez que master tiene varias features, llevamos a cabo una merge request a preproducción con el conjunto de features anteriores, que a su vez, son candidatas de pasar a producción haciendo otro merge request. De esta manera conseguimos que el código subido a producción sea muy estable, ya que validamos featues tanto a nivel individual como en lote.



*Trunk-based Flow:

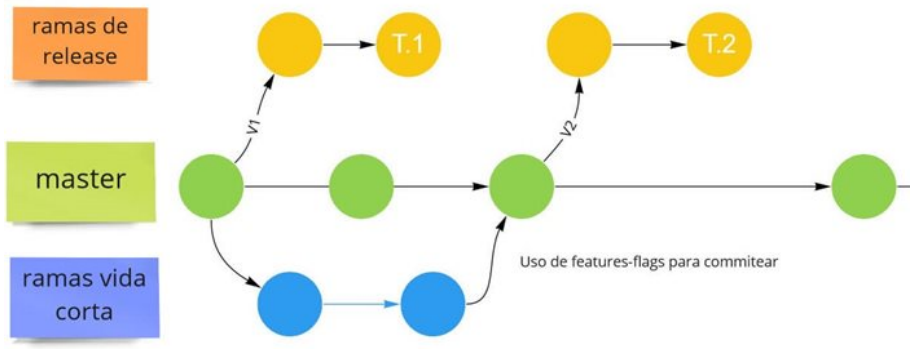
Este flujo es muy similar a GitHub Flow, con la característica nueva de las releases branch y el cambio de filosofía que presenta. Los principios que rigen este flujo son los siguientes:

-Los desarrolladores debemos colaborar siempre sobre el trunk (o master).

-Bajo ningún concepto debemos crear features branch utilizando documentación técnica. En caso de que la evolución sea compleja, haremos uso de features flags (condicionales en el código) para activar o desactivar la nueva característica.

-Preferiblemente usaremos la metodología Pair-Programming en vez de hacer uso de PR.

Se sacan ramas de release para poder desplegar el código en diferentes entornos, ya sea móvil, web, etc.



Parámetros de configuración de las herramientas utilizadas: ¿Qué herramientas de prueba utilizarás y el flujo de trabajo acorde a la Integración continua?

Selenium

Node.js

Node. js es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tienes que preocuparte con el bloqueo de procesos, pues no hay bloqueos.

GitHub ofrece, además de sus funcionalidades, la gestión del código fuente y el control de versiones distribuidas de Git. Sus servicios básicos son gratuitos, y por eso se utiliza para alojar proyectos de código abierto en su mayoría.

2. Entregar un repositorio configurado para recibir el código fuente (liga al repositorio).

https://github.com/CharlieHG/TodoComputacion.git

Referencias Bibliográficas

-🇨🇷 M. Á. (2021, 19 juli). *¿Qué son los feature flags?* DEV Community 👩‍💻👨‍💻. Geraadpleegd op 27 september 2022, van https://dev.to/marianocodes/que-son-los-feature-flags-kc7

-Cinco Git Workflows para mejorar nuestros proyectos. (2021, 30 april). BABEL Sistemas de Información. Geraadpleegd op 27 september 2022, van https://www.babelgroup.com/es/Media/Blog/Abril-2021/Cinco-Git-Workflows-para-mejores-proyectos

-Debe saber la estrategia de implementación en línea que debe conocer - programador clic. (z.d.). Geraadpleegd op 27 september 2022, van https://programmerclick.com/article/5714795477/

-*Just a moment. . .* (z.d.). Geraadpleegd op 27 september 2022, van https://rockcontent.com/es/blog/pruebas-a-b/