INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

MATERIA: GESTIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE

NOMBRE DEL ALUMNO: VICTOR ALDAIR MARTINEZ GAYOSSO

NOMBRE DEL DOCENTE: ING. OSCAR RAFAEL SALAS SALGADO

TRABAJO DE UNIDAD 2

CUATRIMESTRE: 10º

GRUPO: IDGS-101

FECHA: 06 DE MARZO DE 2022

INDICE

INTRODUCCIÓN

En el presente trabajo se explicará el flujo de trabajo (pipeline) para la liberación y despliegue continuo así mismo el entorno requerido para la liberación y despliegue continuo de igual forma los niveles de servicio acordados asi como también las métricas para el monitoreo de la aplicación y los parámetros de configuración de las herramientas utilizadas.

Por consiguiente se describirá las herramienta de liberación continua configurada y vinculada el entorno de despliegue mediante un repositorio configurado: en cual se determinara lo que es el Scripts del flujo de trabajo (pipeline),los Scripts para la generación del entorno de liberación asi mismo Scripts de pruebas en el entorno de liberación y por ultimo los Scripts para la generación del despliegue.

JUSTIFICACIÓN

El Pipeline, que también es conocido como flujo de trabajo, procesos de producción o cadena de producción, se refiere a la organización y administración del paso de la información que existe dentro de una producción de CG (como dije anteriormente, esto puede ser un pipeline de animación, videojuegos, vfx, post-producción, etc).

La manera más sencilla de entenderlo es visualizando la cadena de producción en una fábrica: entra materia prima, esta es modificada por las máquinas y pasa por diferentes procesos para finalmente convertirse en un producto terminado. La importancia de implementar uno dentro de un estudio o compañía, toma mayor importancia conforme los equipos de trabajo se vuelven más grandes y las tareas más complejas. En mi experiencia el pipeline debe responder a las siguientes tres grandes áreas:

Establecer y definir en la cadena de producción los procesos de creación, revisión y aprobación de cada departamento, así como definir los entregables de cada uno de ellos.

Definir los departamentos, los roles de los artistas, las jerarquías de supervisión, dirección y producción, así como sus responsabilidades y adjudicaciones.

Definir las nomenclaturas, la organización de archivos, la seguridad y control de versiones, implementar las buenas prácticas de trabajo, identificar cuellos de botella y fomentar la automatización de procesos.

El objetivo es que el pipeline sea la grasa que permita a toda la maquinaria de producción estar bien lubricada para que esta funcione eficientemente, sin contratiempos, de manera previsible y con un mínimo de downtime o tiempos muertos, así como de mantener la misma calidad.

I. JUSTIFICACIÓN DEL FLUJO DE TRABAJO (PIPELINE) PARA LA LIBERACIÓN Y DESPLIEGUE CONTINUO.

El Pipeline, que también es conocido como flujo de trabajo, procesos de producción o cadena de producción, se refiere a la organización y administración del paso de la información que existe dentro de una producción de CG (como dije anteriormente, esto puede ser un pipeline de animación, videojuegos, vfx, postproducción, etc). La manera más sencilla de entenderlo es visualizando la cadena de producción en una fábrica: entra materia prima, esta es modificada por las máquinas y pasa por diferentes procesos para finalmente convertirse en un producto terminado.

La importancia de implementar uno dentro de un estudio o compañía, toma mayor importancia conforme los equipos de trabajo se vuelven más grandes y las tareas más complejas. En mi experiencia el pipeline debe responder a las siguientes tres grandes áreas:

- Establecer y definir en la cadena de producción los procesos de creación, revisión y aprobación de cada departamento, así como definir los entregables de cada uno de ellos.
- Definir los departamentos, los roles de los artistas, las jerarquías de supervisión, dirección y producción, así como sus responsabilidades y adjudicaciones.
- Definir las nomenclaturas, la organización de archivos, la seguridad y control de versiones, implementar las buenas prácticas de trabajo, identificar cuellos de botella y fomentar la automatización de procesos.

El despliegue continuo o CD (continuous deployment) está relacionado estrechamente con la entrega continua. Con el despliegue continuo se va un paso más allá de la entrega continua, automatizando todo el proceso de entrega de software al usuario, eliminando la acción manual o intervención humana necesaria en la entrega continua, El despliegue continuo libera de carga a los equipos de

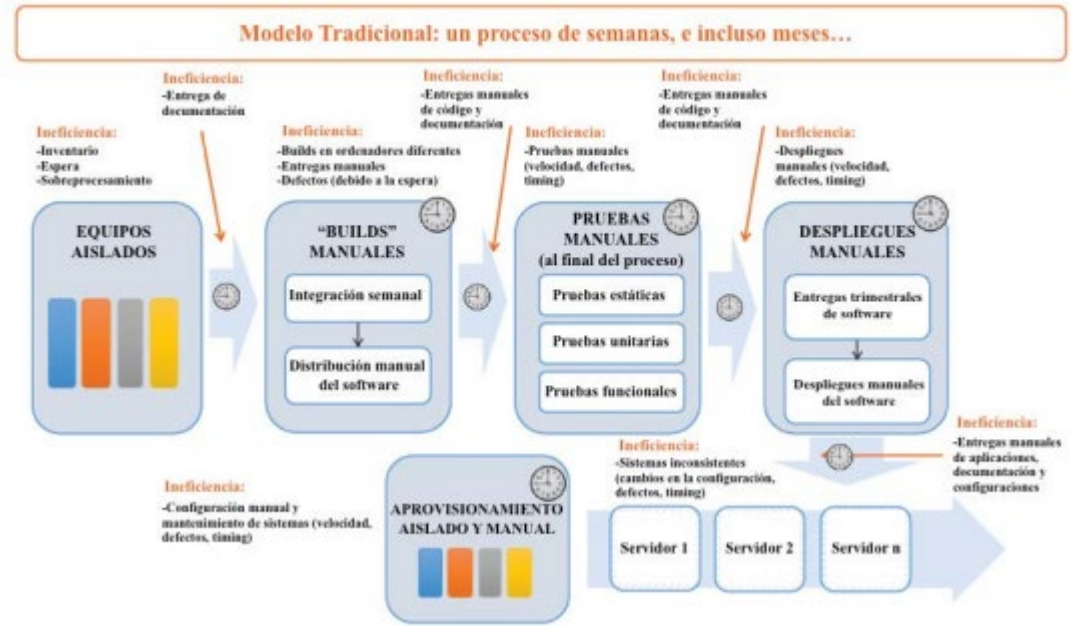
operaciones de procesos manuales, que son una de la principal causa de retrasos en la distribución de aplicaciones.

1. LIBERACIÓN CONTINUA

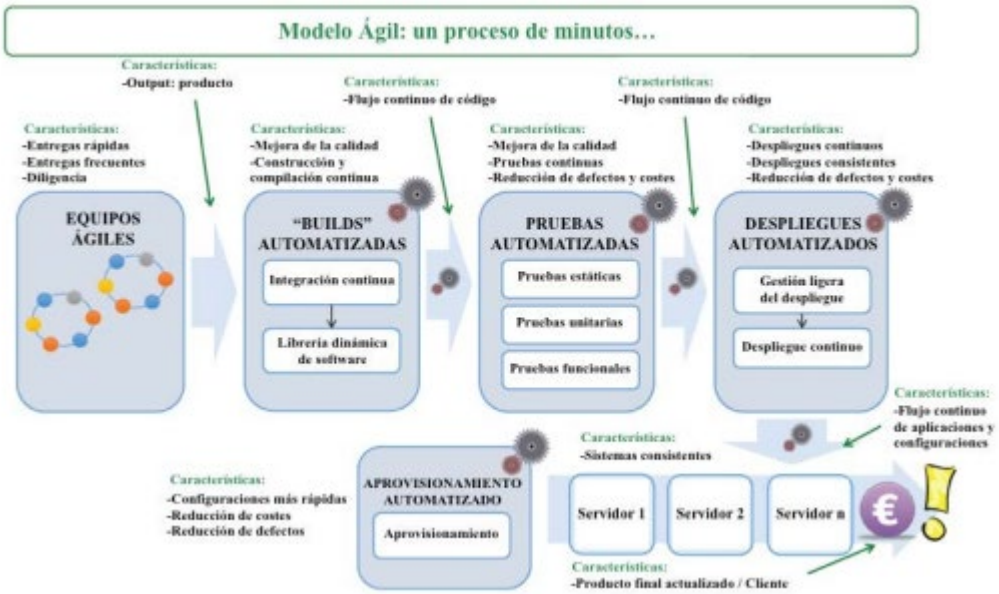
En el desarrollo de software, el modelo de lanzamiento continuo, liberación continua, actualización rodante o actualización continua se refiere a un sistema de lanzamiento y actualizaciones constante del software. Esto, en contraste con un modelo de desarrollo estándar de liberación que utiliza diferentes versiones que deben reinstalarse sobre la versión anterior, genera un menor riesgo al actualizar componentes importantes del programa, ya que son actualizados gradualmente.

2. ENTORNO REQUERIDO PARA LA LIBERACIÓN Y DESPLIEGUE CONTINUO.

El primer modelo de despliegue de aplicaciones que se explicará será el Modelo Tradicional, caracterizado, sobre todo, por la leve o nula automatización de todos sus procesos. Para la explicación de las características de este modelo, se disgregará en cuatro partes, que representan el flujo de trabajo en cualquier proyecto de ingeniería de software, y que serán fundamentales a la hora de realizar una comparación posterior entre ellos: equipos, construcción de código, pruebas y despliegue. En la figura 1 se puede ver, de forma gráfica, un resumen del flujo de trabajo en el Modelo Tradicional, con todas las partes que se describirán a continuación.



Modelo Ágil de despliegue de aplicaciones El siguiente modelo de despliegue de aplicaciones que se explicará será el Modelo Ágil, caracterizado, sobre todo, automatización de todos sus procesos. Para la explicación de las características de este modelo, al igual que para la del Modelo Tradicional, se disgregará en las cuatro partes correspondientes al flujo de trabajo en cualquier proyecto de ingeniería de software. En la figura 2 se puede ver, de forma gráfica, un resumen del flujo de trabajo en el Modelo Ágil, con todas las partes que se describirán a continuación.



3. **NIVELES DE SERVICIO ACORDADOS.**

Un acuerdo de nivel de servicio (SLA) es un contrato que establece un conjunto de entregables que una parte ha acordado proporcionar a la otra. Este acuerdo puede existir entre una empresa y sus clientes, o un departamento que ofrece un servicio recurrente a otro departamento dentro de la empresa.

Los SLA son comunes para un negocio cuando se registran nuevos clientes. Cuando existe uno entre los departamentos de ventas y marketing, este acuerdo detalla los objetivos de marketing, como el número de leads o la canalización de ingresos; y las actividades de ventas que los seguirán y respaldarán, como la atracción de leads que fueron calificados por el equipo de marketing.

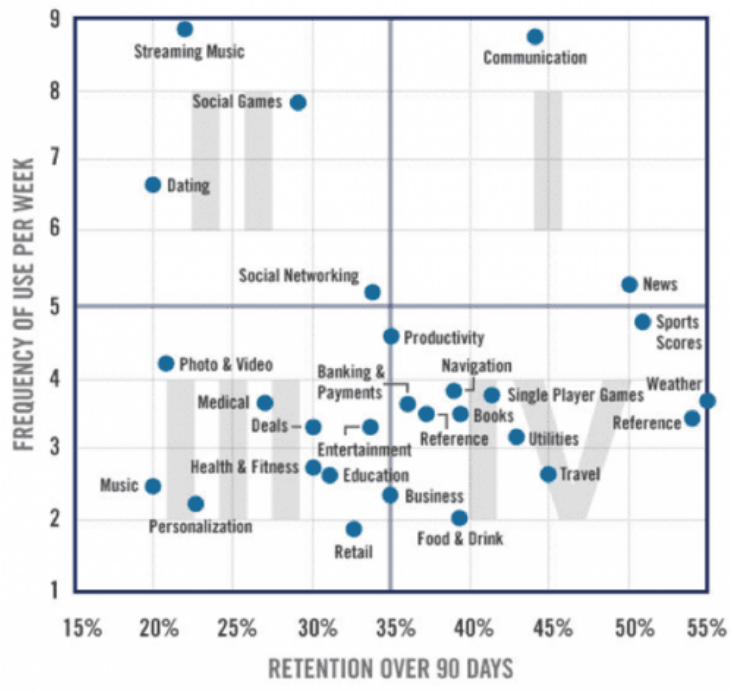
Tanto el departamento de ventas como el de marketing utilizan este documento como un compromiso para apoyarse mutuamente, basándose en objetivos numéricos concretos. ¿Y adivina qué? El 65% de los especialistas en marketing

cuyas empresas tienen este tipo de SLA ven un mayor retorno de la inversión en sus esfuerzos de inbound marketing.

Acuerdo de nivel de servicio

Marketing	Ventas
Punto de contacto: *Nombre* *Email*	Punto de contacto: *Nombre* *Email*
Objetivos	
Tráfico: *meta de tráfico* Leads: *meta de leads* MQL: *meta de MQL*	Oportunidades/demos: *meta op/demo* Tratos: *meta de tratos* Ingresos: *meta de ingresos*
Iniciativas para (periodo)	
En [periodo], marketing se enfocará en: • Iniciativa 1 • Iniciativa 2 • Iniciativa 3	En [periodo], ventas se enfocará en: • Iniciativa 1 • Iniciativa 2 • Iniciativa 3

4. MÉTRICAS PARA EL MONITOREO DE LA APLICACIÓN.



- **Usuarios Activos**

Descargar una aplicación es la parte más fácil. Conseguir que el usuario vuelva a entrar ya cuesta un poco más. Es por esta razón que debemos considerar tanto los usuarios activos mensuales (MAU) como los diarios (DAU). Conociéndolo todo sobre ellos, el uso que hacen de la aplicación, de dónde son, etc te permite segmentarlos y definir acciones personalizadas a cada uno con el objetivo de fidelizarlos.

Los usuarios activos son importantes, pero también nuestra capacidad de atraer nuevos usuarios. El porcentaje de nuevos usuarios es otra métrica para apps que no debemos olvidar: ¡cada vez que disminuye este porcentaje debe sonar nuestra alarma!

- **Uso de la aplicación**

Una de mis preguntas preferidas es: ¿en qué pantalla pierdes la mayoría de tus usuarios? ¿Lo sabéis? ¡Es un dato clave! Conocer el flujo de navegación dentro de la aplicación es muy importante ya que nos permite saber por qué pantallas han ido nuestros usuarios y, sobre todo, en qué pantallas se han quedado. En un juego este dato nos puede informar sobre un nivel demasiado difícil. En una app sobre productividad, nos informará de algún error de planteamiento.

Otra cuestión a tener en cuenta es la navegación dentro de cada pantalla para saber si realmente los usuarios están siguiendo el proceso que nosotros habíamos pensado. Para ellos los heat maps o mapas de calor nos pueden ayudar.

- **Largada de la sesión**

¿Cuántos minutos tiempo utilizan la aplicación? Los que entran y salen, seguramente se han encontrado con una app que no es lo que buscaban. Si esta es la mayoría, estamos delante de una incorrecta comunicación (puede que en nuestra web y en el store).

¿Cuánto tiempo tienes valorado que utilicen la app? Si por ejemplo se trata de una app sobre la meteorología, posiblemente tendrá sentido que la mayoría de personas la utilicen entre 1 y 10 minutos. En cambio, si se trata de una app de productividad, posiblemente nos interese que la utilicen más de 30 minutos.

- **Retención**

El 20% de las aplicaciones móviles sólo se utilizan una vez. ¿Lo sabías? Tenemos que intentar que nuestra app no sea una de estas. ¿Cómo detectarlo? Podemos medir la retención como el porcentaje de usuarios que vuelven a la app después de su primera visita. Y no solamente es importante saber la retención de los usuarios, sino que también la frecuencia con la que vuelven a a la app. Como más comprometidos y fidelizados sean nuestros usuarios, mejores estrategias de monetización podremos desarrollar.

- **Coste de adquisición del cliente (CAC)**

Saber cuánto nos cuesta adquirir un usuario es importante ya que en función de este dato sabremos hasta cuánto podemos invertir en publicidad. El CAC se calcula sumando todos los gastos empleados en conseguir el nuevo cliente (marketing, comercial, infrastructura) y se divide por el número de clientes conseguidos en este mismo periodo.

CAC = Gastos necesarios para captar un cliente / Nuevos clientes

- **Ingresos medios por cliente o average revenue per user (ARPU)**

Es muy importante que los usuarios utilicen nuestra aplicación, que estén comprometidos y vuelvan a ella. Tan importante como esto es saber rentabilizarlos. Para ello utilizamos el ARPU que se calcula sumando todos los

ingresos por usuario (precio de la app, in-app purchases, anuncios, etc) y se divide por el número de usuarios.

- **Vida útil o customer lifetime value (LTV)**

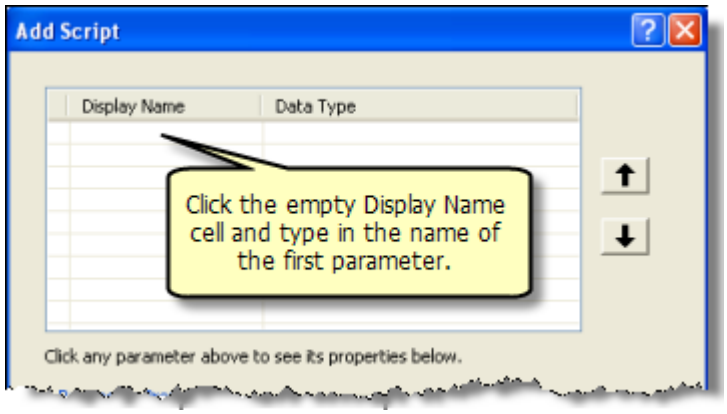
¿Quieres saber cuál es el valor del usuario? Es el resultado de multiplicar el ARPU por la vida media que se espera de un cliente. Teniendo este dato podremos hacer previsiones sobre lanzamientos y necesidades de capital.

5. PARÁMETROS DE CONFIGURACIÓN DE LAS HERRAMIENTAS UTILIZADAS.

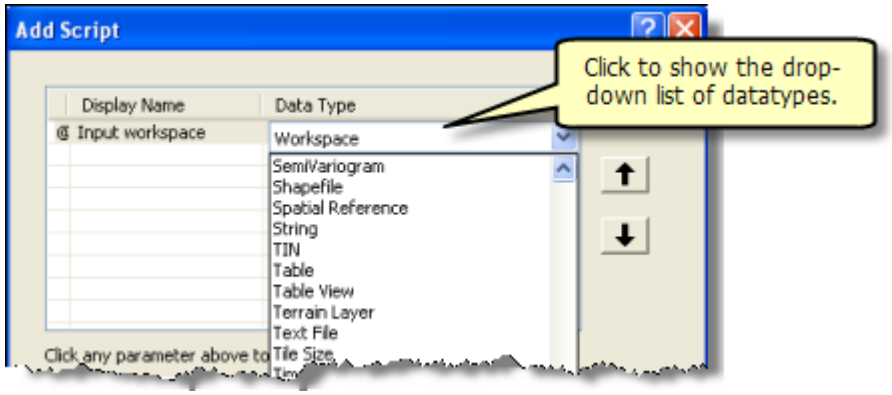
Los parámetros de una herramienta de secuencia de comandos se pueden establecer mediante el asistente Agregar secuencia de comandos. También puede agregar, eliminar y modificar los parámetros de una herramienta de secuencia de comandos en el cuadro de diálogo Propiedades de la herramienta. Para acceder a las propiedades de la herramienta de secuencia de comandos, haga clic con el botón derecho del ratón en la herramienta, después haga clic en Propiedades y en la pestaña Parámetros.

Ya sea que esté configurando los parámetros en el asistente Agregar secuencia de comandos o en el cuadro de diálogo Propiedades, los procedimientos (como se describe aquí) son iguales.

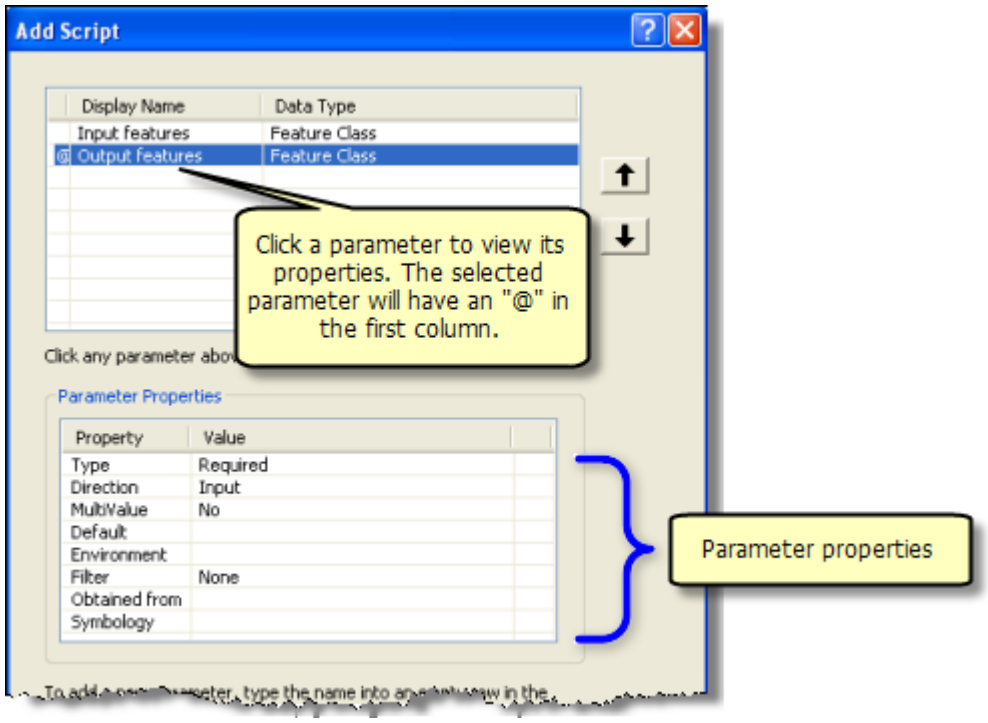
Para agregar un parámetro nuevo, haga clic en la primera celda vacía de la columna Nombre de visualización y escriba el nombre del parámetro. Este es el nombre que se visualizará en el cuadro de diálogo de la herramienta y puede contener espacios. El nombre del parámetro para la sintaxis de Python será el nombre de visualización con los espacios reemplazados por guiones bajos (_).



Después de escribir el nombre de visualización del parámetro, seleccione un tipo de datos para el parámetro haciendo clic en la celda **Tipo de datos**, como se muestra a continuación.



Cada parámetro tiene propiedades adicionales que usted puede configurar, como se muestra a continuación.



6. PROPIEDADES DE LOS PARÁMETROS.

6.1. Tipo

Existen tres opciones de Tipo:

- Un parámetro Requerido requiere que el usuario introduzca un valor de entrada. No se puede ejecutar la herramienta hasta que el usuario suministre un valor.
- Un parámetro Opcional no requiere que el usuario introduzca un valor.
- Un parámetro Derivado sólo se utiliza con parámetros de salida (consulte Dirección a continuación). Un parámetro de salida derivada no se muestra en el cuadro de diálogo de la herramienta.

Un parámetro de salida derivada tiene los siguientes cinco usos:

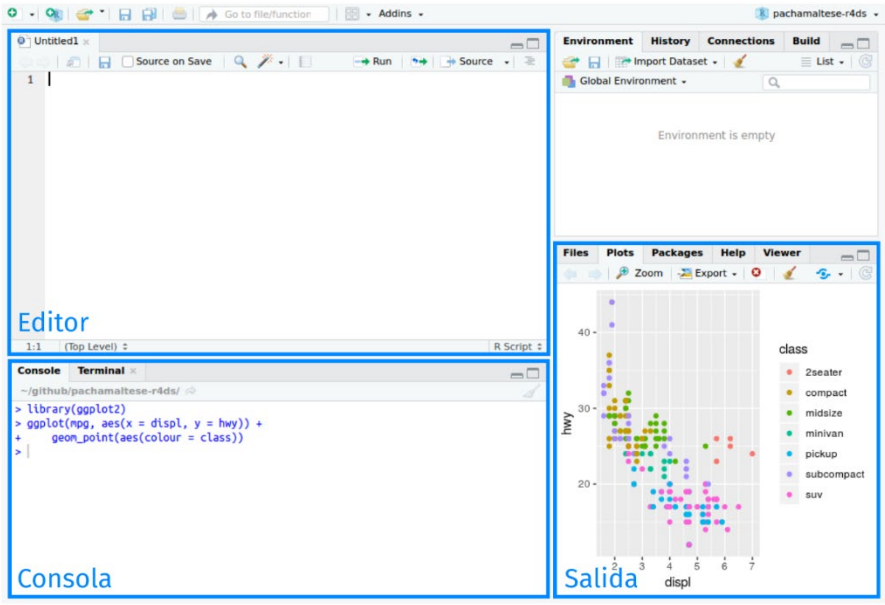
  - La salida es la misma que la entrada, como en Calcular campo. Calcular campo cambia los valores de un campo en particular en la tabla de entrada; no crea una tabla nueva ni modifica el esquema

de la entrada. Se pueden encontrar otras herramientas en las cuales la salida es la misma que la entrada en la caja de herramientas Edición.

- La herramienta modifica el esquema de la entrada, como en Agregar campo. Agregar campo agrega un campo a la tabla de entrada; no crea una tabla de salida nueva.
- La herramienta utiliza la información en otros parámetros para crear una salida, como la herramienta Crear clase de entidad. Con la herramienta Crear clase de entidad, puede especificar el espacio de trabajo y el nombre de la nueva clase de entidad, y se crea la clase de entidad por usted.
- La herramienta genera como salida un valor escalar, a diferencia de un dataset. Obtener recuento, por ejemplo, obtiene un número entero largo (la cantidad de registros). Siempre que la herramienta obtenga un valor escalar, la salida es Derivado.
- La herramienta creará los datos en una ubicación conocida. Por ejemplo, puede tener una secuencia de comandos que actualice una tabla existente en un espacio de trabajo conocido. No es necesario que el usuario suministre esta tabla en el cuadro de diálogo o en la secuencia de comandos.

7. FLUJO DE TRABAJO: SCRIPTS

Para tener más espacio de trabajo, una buena idea es usar el editor de script. Ábrelo ya sea haciendo clic en el menú de Archivo (File), seleccionando Nuevo Archivo (New File), y luego Script de R (R Script), o bien, utilizando el atajo del teclado Cmd/Ctrl + Shift + N. Ahora verás cuatro paneles:



El editor de script es un excelente espacio para colocar el código que te importa. Continúa experimentando en la consola, pero una vez que has escrito un código que funciona y hace lo que quieres, colócalo en el editor de script. RsStudio guardará automáticamente los contenidos del editor cuando salgas del programa, y los cargará automáticamente cuando vuelvas a abrirlo. De todas formas, es una buena idea que guardes los scripts regularmente y que los respaldes.

Ejecutando código

El editor de *script* es también un excelente espacio para desarrollar gráficos de **ggplot2** complejos o largas secuencias de manipulación con **dplyr**. La clave para utilizar el editor de *script* efectivamente es memorizar uno de los atajos del teclado más importantes: Cmd/Ctrl + Enter. Esto ejecuta la expresión actual de R en la consola. Por ejemplo, toma el código de abajo. Si tu cursor está sobre no_cancelado <- vuelos %>%, presionar Cmd/Ctrl + Enter ejecutará el comando completo que genera no_cancelado. También moverá el cursor al siguiente enunciado (que comienza con no_cancelado %>%). Esto facilita ejecutar todo tu *script* presionando repetidamente Cmd/Ctrl + Enter.

```
library(dplyr)
library(datos)
no_cancelado <- vuelos %>%
  filter(!is.na(atraso_salida), !is.na(atraso_llegada))
no_cancelado %>%
  group_by(anio, mes, dia) %>%
  summarise(media = mean(atraso_salida))
```

En lugar de correr expresión por expresión, también puedes ejecutar el *script* completo en un paso: Cmd/Ctrl + Shift + S. Hacer esto regularmente es una buena forma de verificar que has capturado todas las partes importantes de tu código en el *script*. Te recomendamos que siempre comiences tu *script* cargando los paquetes que necesitas. De este modo, si compartes tu código con otras personas, quienes lo utilicen pueden fácilmente ver qué paquetes necesitan instalar. Ten en cuenta, sin embargo, que nunca deberías incluir install.packages() (del inglés, *instalar paquetes*) o setwd() (del inglés *set working directory*, establecer directorio de trabajo) en un *script* que compartes. ¡Es muy antisocial cambiar la configuración en la computadora de otra persona!

Diagnósticos de RStudio

El editor de script resaltará errores de sintaxis con una línea roja serpenteante bajo el código y una cruz en la barra lateral:

```
4  x y <- 10
```

Sitúate sobre la cruz para ver cuál es el problema:

```
4  x y <- 10
   unexpected token 'y'
   unexpected token '<-'
```

RStudio te informará también sobre posibles problemas:

```
17  3 == NA
    use 'is.na' to check whether expression evaluates to
     NA
```

CONCLUSIÓN

Como conclusión se deduce que un pipeline no está restringido a las empresas únicamente. Muchos artistas -consciente o inconscientemente- desarrollan sus propios flujos de trabajo: la administración de la información, el repositorio y nombrado de archivos, la organización de información para transferir los assets digitales entre programas, los scripts de automatización que se hacen/usan para ciertos procesos etc.

BIBLIOGRAFÍA

¿Qué es un Pipeline? Una Introducción al Pipeline en CG. (2020, julio 9). IndustriaAnimacion.com. https://www.industriaanimacion.com/2020/07/que-es-pipeline-una-introduccion-pipeline-cg/

Aguilar, Q. (2020, diciembre 10). Integración continua, entrega continua y despliegue continuo. Ilimit.com. https://www.ilimit.com/blog/integracion-continua-entrega-continua-despliegue-continuo/

N. Rathod and A. Surve, “Test orchestration a framework for Continuous Integration and Continuous deployment,” in 2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015, 2015.

D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, EMF: eclipse modeling framework. 2008.