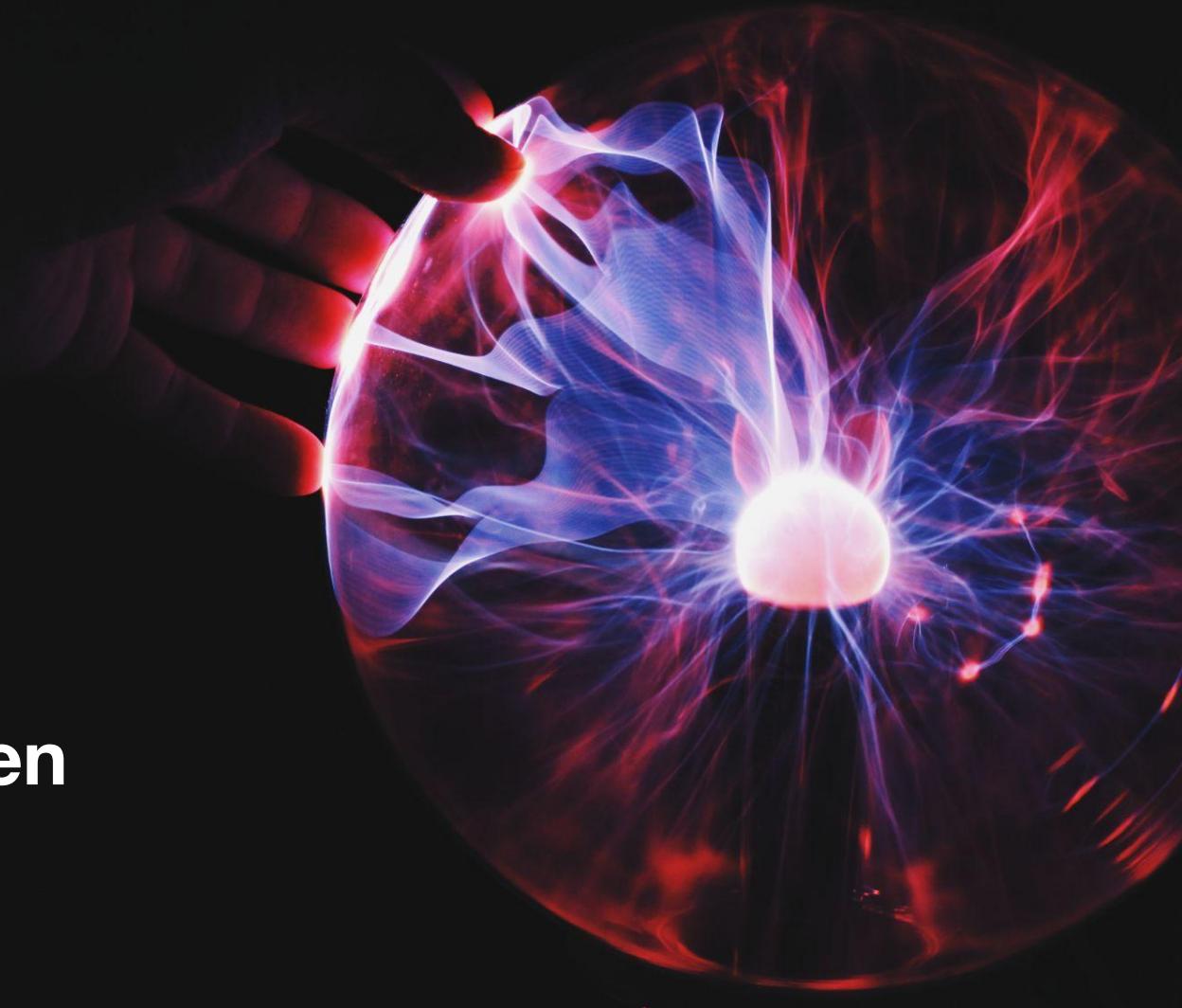


INPUT | OUTPUT

De UTxO a Aiken

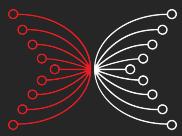
Azteca - 2025



¡Hola!

Gracias por estar aquí y por tu entusiasmo por aprender sobre Cardano.





Karina Lopez

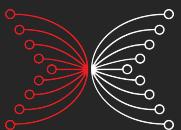
Instructor



Antonio Ibarra

Instructor

Conoce al equipo de educación de IO



Dr. Lars Brünjes

Director of Education



Niamh Ahern

Education Manager



Alejandro Garcia

Education Lead -
Agile



Karina Lopez

Education Lead -
Technical



Dr. Arturo Mora

Education Lead -
Curriculum



Robertino Martinez

Education Lead -
Delivery



Antonio Ibarra

Senior Education Specialist



Tuvshintsenguu Erdenejargal

Senior Education Specialist



Luka Kurnjek

Education Specialist



Dr. Alexey Sorokin

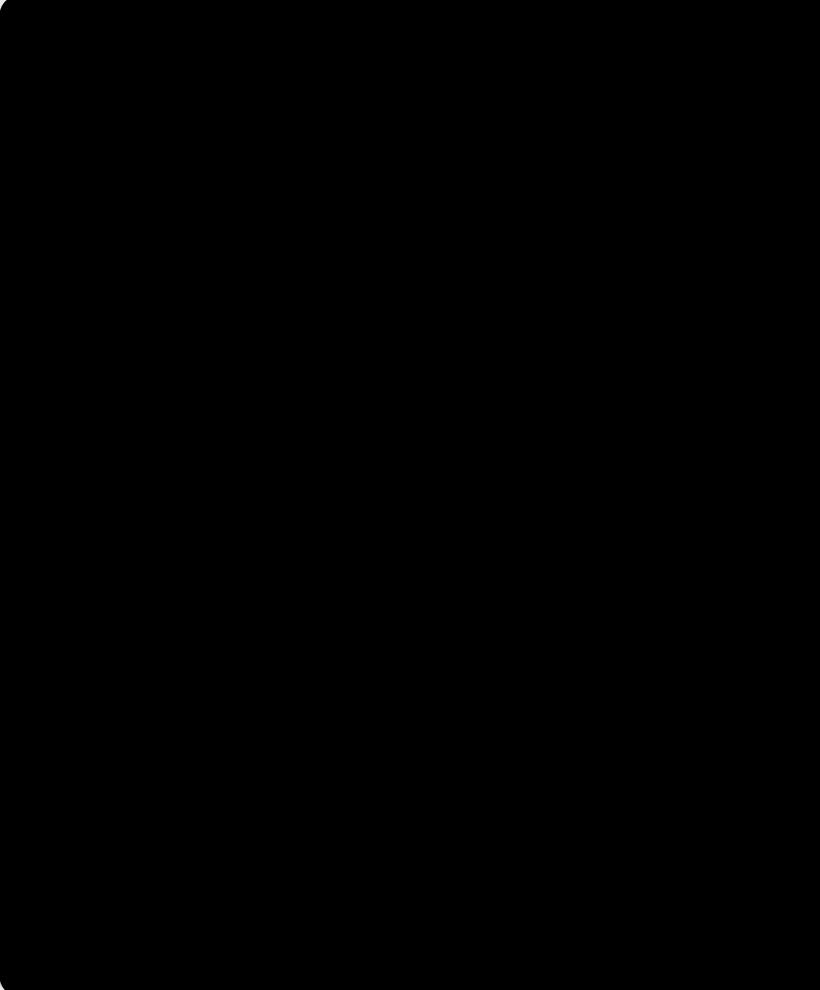
Education Specialist

**Levanten la
mano si alguna
vez han
escuchado las
palabras
contrato
inteligente**



Objetivo del workshop

El Cardano Developer Workshop (CDW) es un workshop proveído por el equipo de educación de IOG y tiene como **principal objetivo introducir desarrolladores de Smart Contracts y aplicaciones distribuidas (DApps) en Cardano.**



↗ Contenido

- Resumen Blockchain
- Modelos: Contables, UTxO y Extendido (EUTxO)
- ¿Qué son los Contratos Inteligentes?
- Introducción a Aiken
- Programación con Aiken

+ Blockchain _

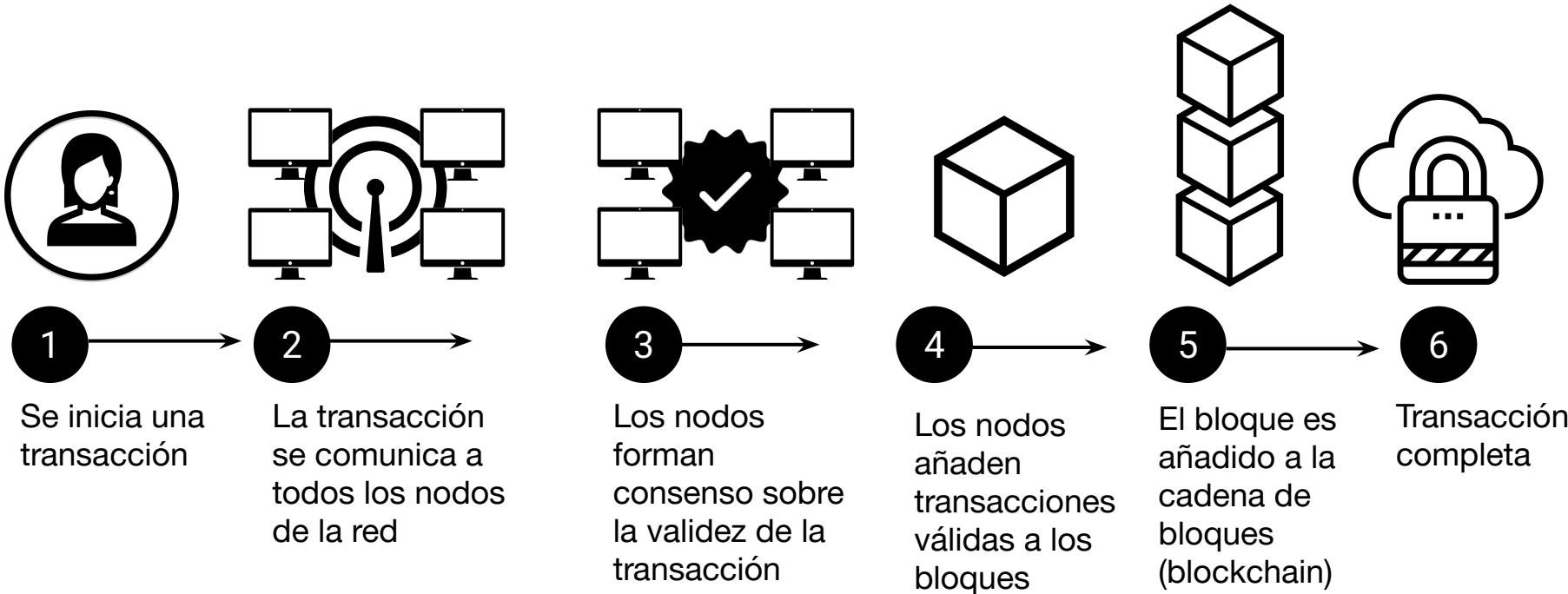


↗ Blockchain

- La Blockchain se puede comparar con un libro de registros permanentes.
- Descentralizada.
- Solo escritura.

Blockchain: Base de Datos Descentralizada

La base de datos se sincroniza a través de la red, con reglas especiales que se definen para incentivar a los buenos actores y desincentivar a los malos.



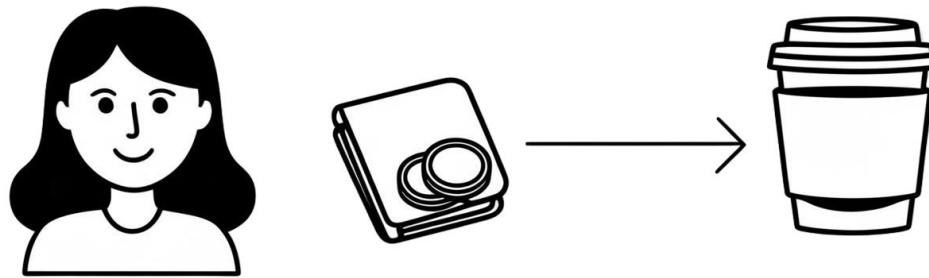
**¿Alguna vez te haz preguntado
cómo hacer programable la
blockchain de Cardano?**

+ Modelos _

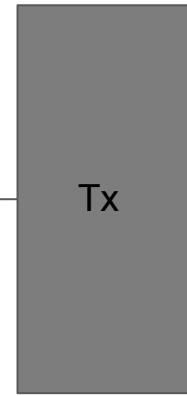
12

Alicia compra un café

- Alicia comienza su día con un plan para comprar un café en su cafetería favorita.



Transacciones en efectivo



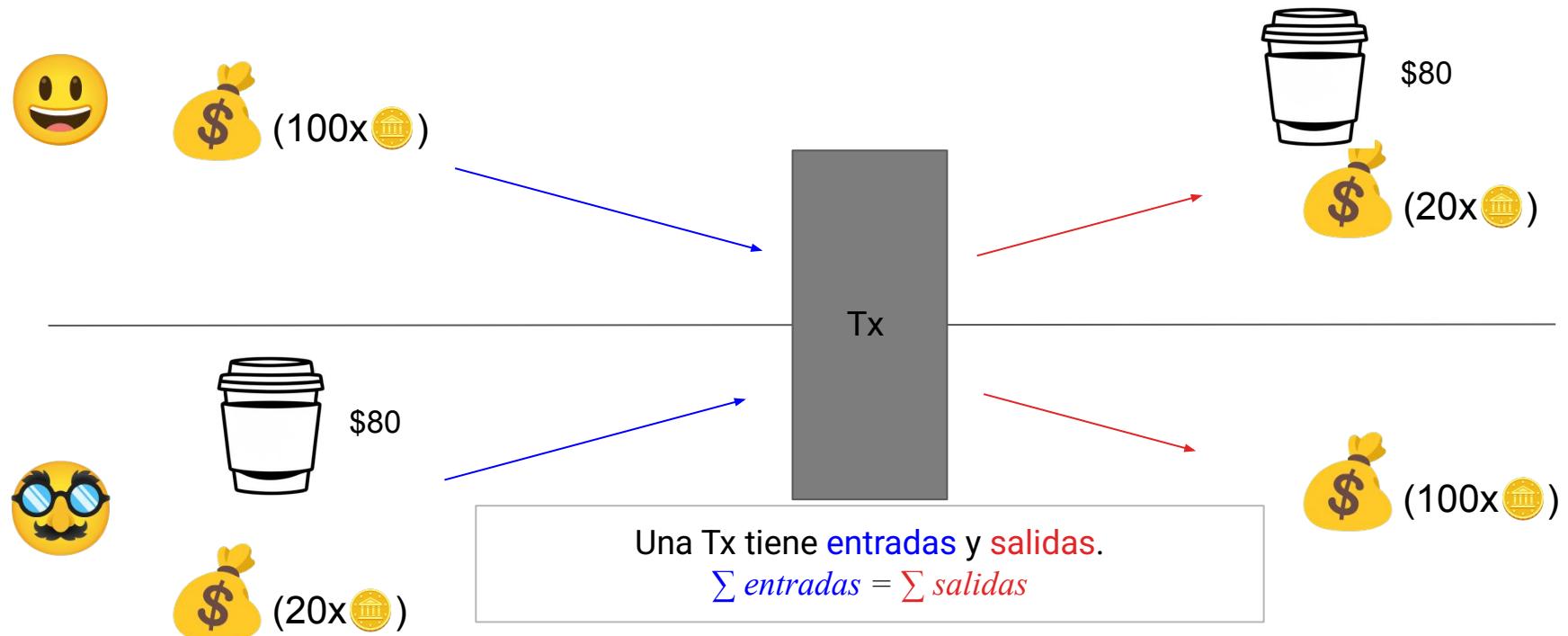
\$80



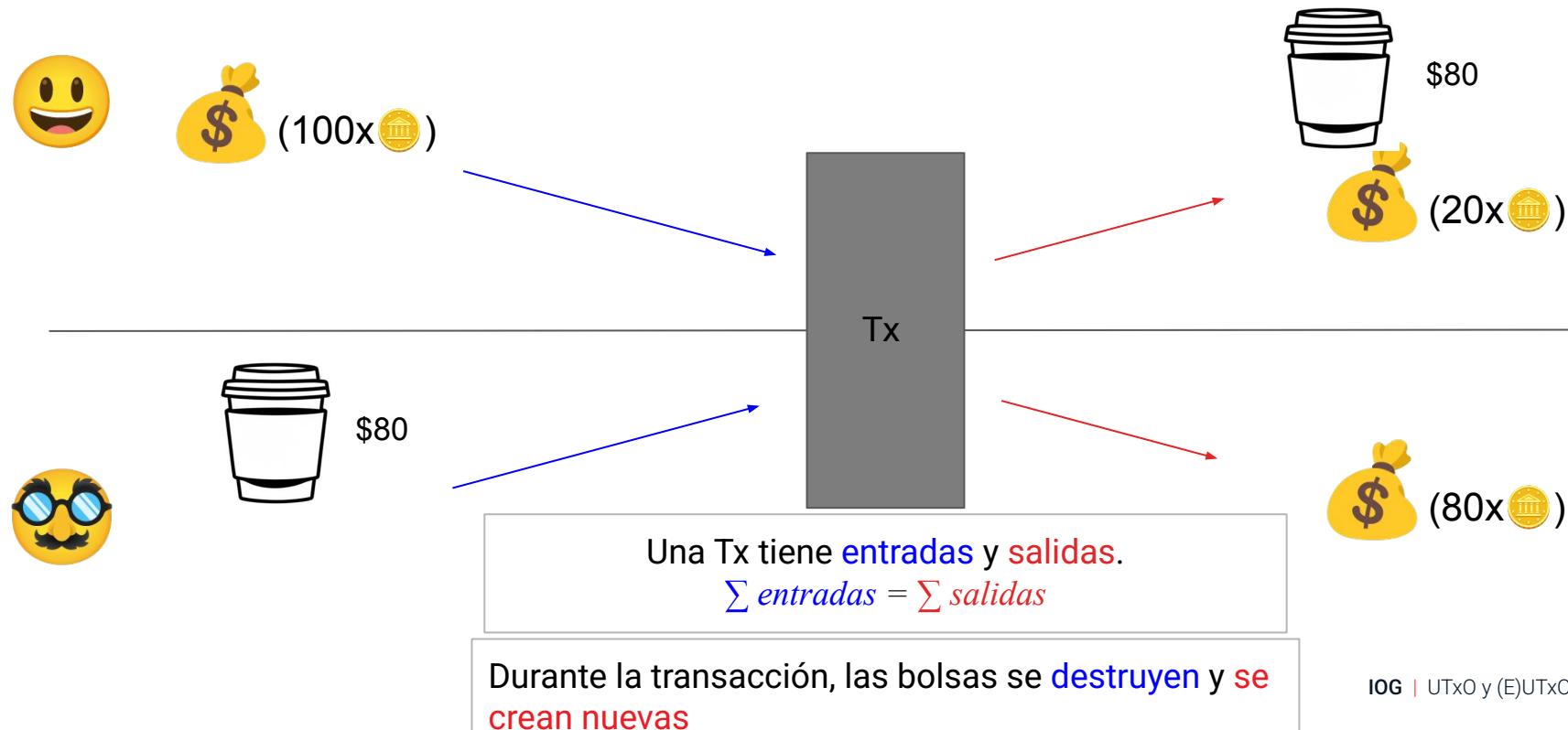
Tx



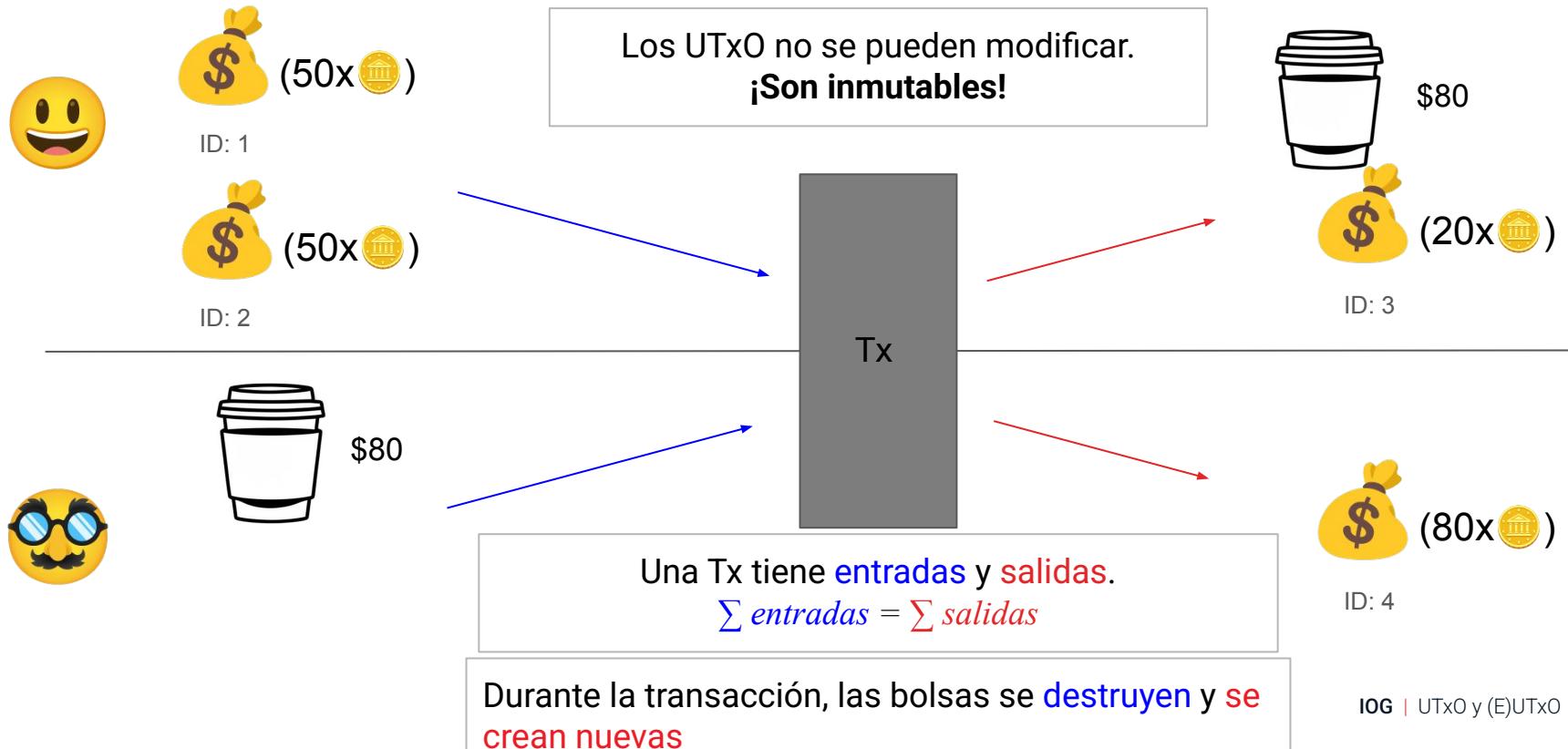
Transacciones UTxO



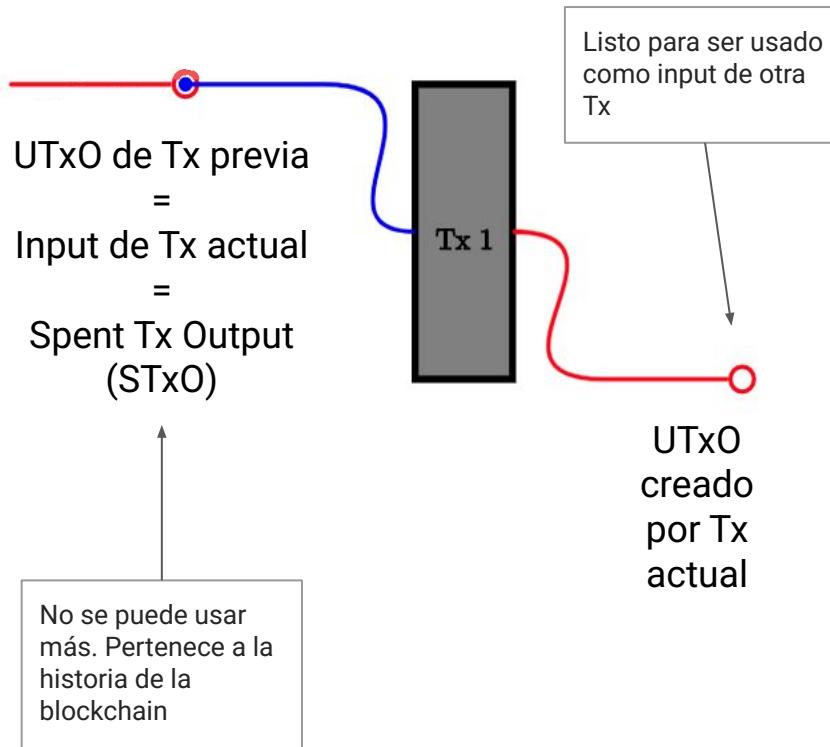
Transacciones UTxO



Transacciones UTxO



Modelo UTxO

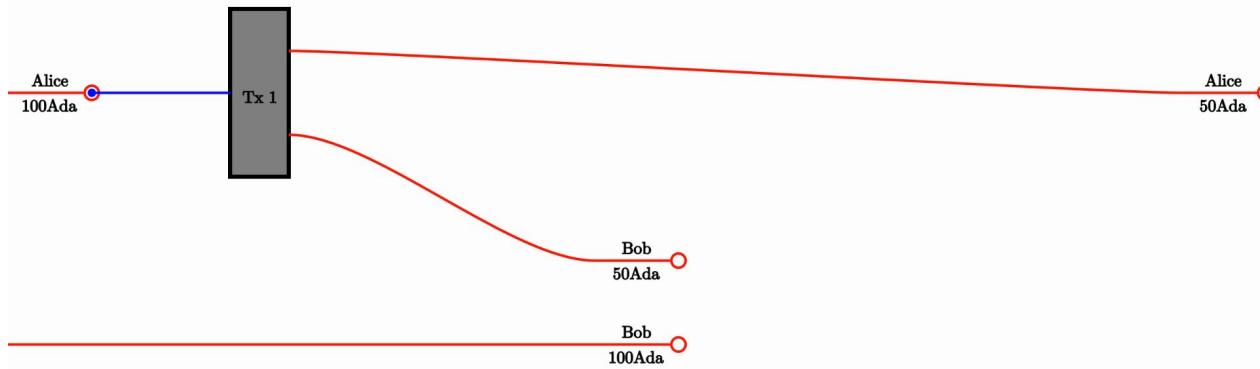


- Transacciones tienen un **número arbitrario de inputs** (entradas) y outputs (salidas). (Mínimo 1 y 1).
- **Unspent Transaction Output (UTxO)** son las salidas de transacciones que todavía no han sido consumidas por otras transacciones.
- Si se quiere usar parte del valor encerrado en un UTxO, se tiene que **consumir completamente**.
- Lo único que pasa en la blockchain son estas transacciones. Los UTxO se consumen y crean, pero **NO SE MODIFICAN!**

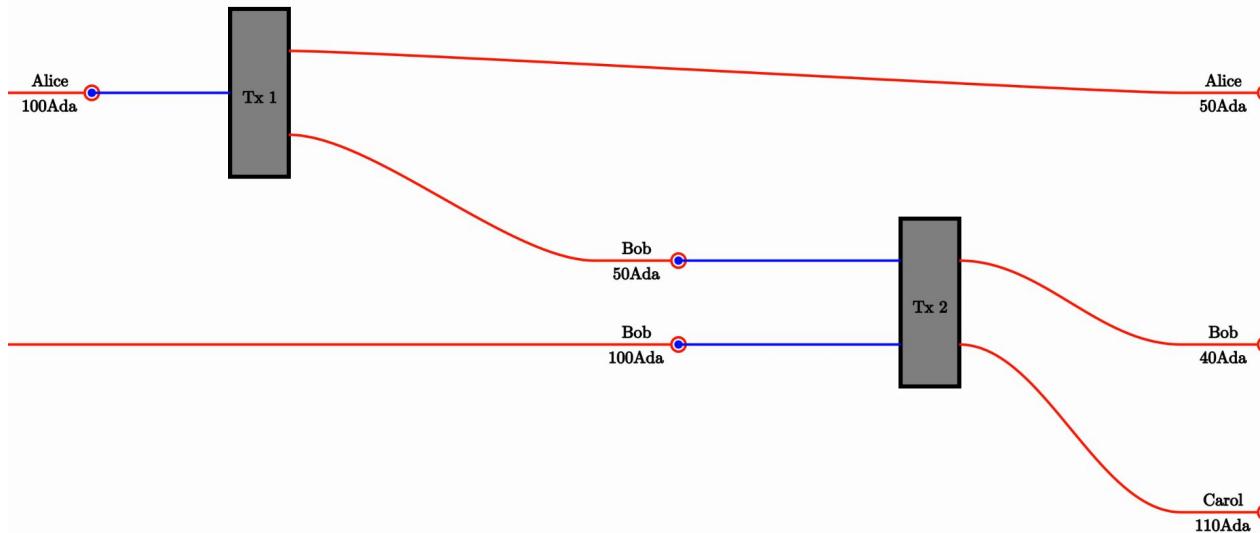
Transacciones UTxO: 🖊 Alice le manda 50 ADA a Bob?



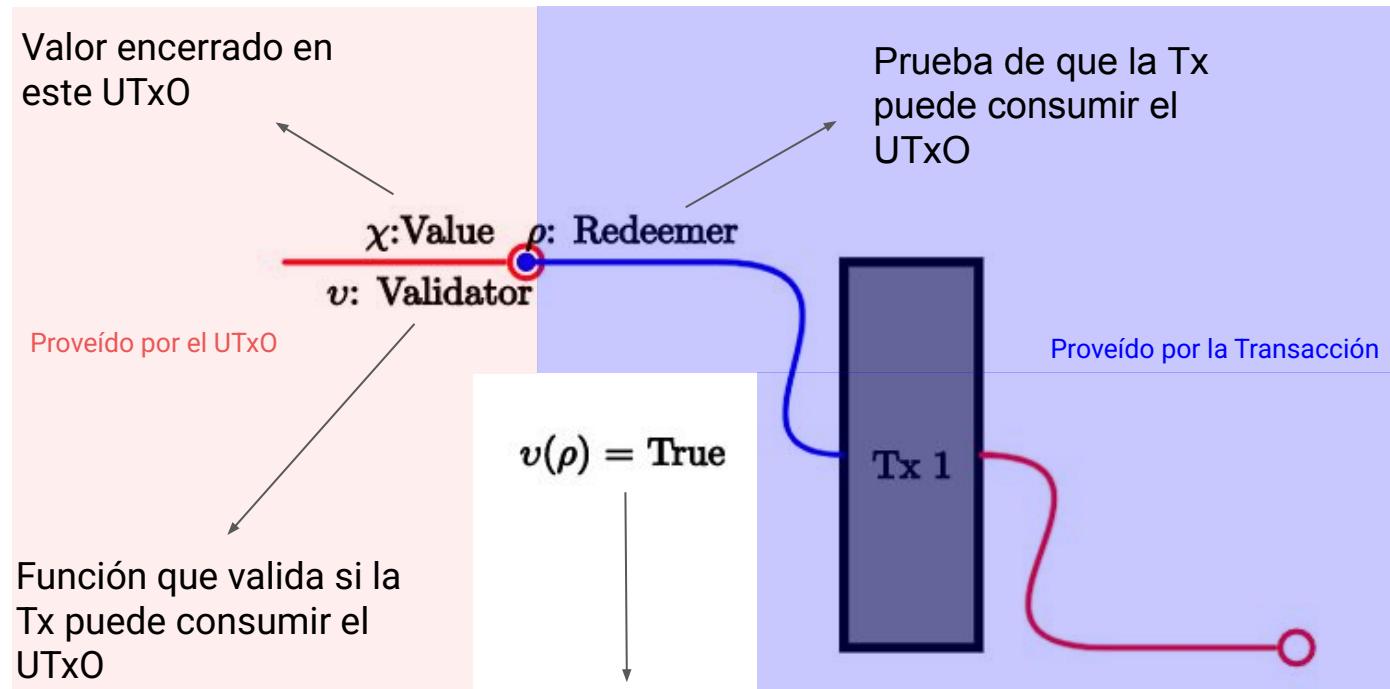
Transacciones UTxO: 🖊 Alice le manda 50 ADA a Bob?



Transacciones UTxO: ✎ ¿Bob le manda 110 Ada a Carol?



Transacciones en UTxO: Cómo me aseguro que no consuman mis UTxO?



Se aplica el Validador al Redeemer y devuelve **True/False** si la Tx puede/no puede consumir el UTxO

Transacciones en UTxO: Limitaciones del modelo UTxO

El Modelo UTxO sirve perfectamente para **transferencias simples**. Pero **no tiene suficiente información** como para realizar transferencias más complejas. Por ejemplo:

- Permitir consumir el UTxO después de una fecha específica
- Permitir consumir el UTxO sólo si se envía a una persona específica
- Permitir consumir el UTxO sólo si también se consume otro UTxO específico
- Permitir consumir el UTxO sólo si existe otro UTxO (que no es consumido) con un valor específico
- ...

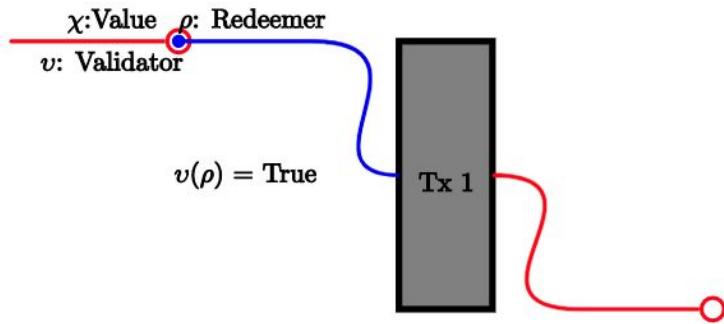
Pero esas son cosas **necesarias** para poder crear **programas descentralizados** que permitan hacer **lo mismo** que hoy en día se utiliza **centralizadamente**:

- Prestamos
- Financiaciones
- Inversiones
- Calendario de adquisición de acciones
- ...

+ MODELO (E)UTxO _

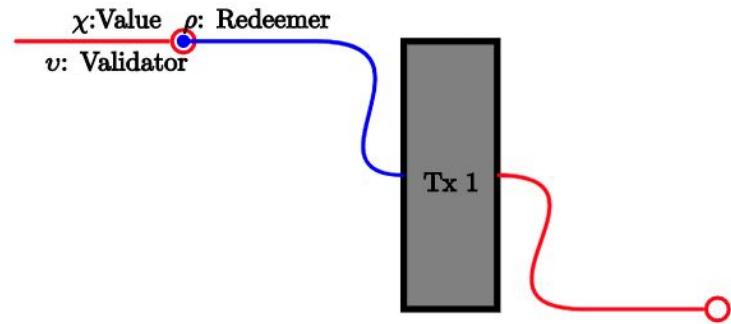
(E)UTxO: UTxO vs (E)UTxO

UTxO



Solo necesitamos nuestra llave privada para gastar nuestros activos.

UTxO Extendido



Es capaz de ejecutar Contratos Inteligentes en Cardano, permitiéndonos establecer reglas para gastar nuestros activos.

+ ¿Qué son los Contratos Inteligentes? —

Programabilidad y Contratos Inteligentes

Cardano soporta contratos inteligentes programables y aplicaciones descentralizadas (DApps).

La introducción de contratos inteligentes en la era Goguen permite a los desarrolladores crear una variedad de aplicaciones descentralizadas para diversos casos de uso.

Los contratos inteligentes en Cardano permiten aplicaciones como:

- Mercados y plataformas NFT
- Intercambios descentralizados (DEX)
- Plataformas automatizadas de préstamos y créditos
- Plataformas de gestión de identidad digital
- Red móvil descentralizada impulsada por blockchain
- Sistemas descentralizados de inteligencia artificial
- Organizaciones autónomas descentralizadas (DAO)
- Mercados de predicción descentralizados
- Sistemas descentralizados de almacenamiento en la nube

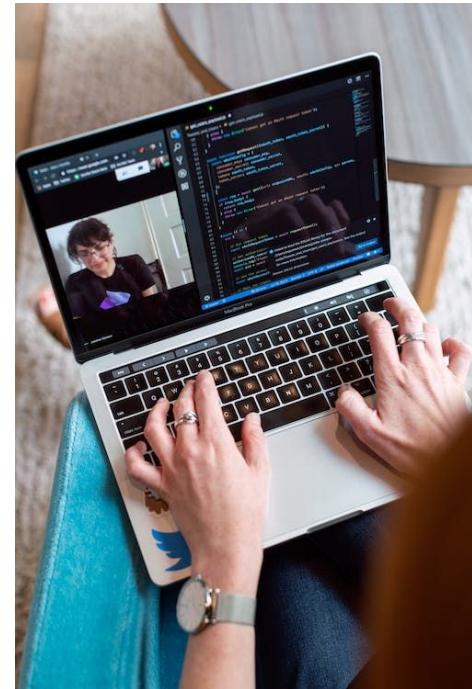


Photo by ThisIsEngineering from [Pexels](#)

(E)UTxO: Contratos Inteligentes en acción

Alice
100 ADA

Digamos que Alice tiene 100 ADAs y Bob tiene 25 ADAs

Bob
25 ADA

(E)UTxO: Contratos Inteligentes en acción

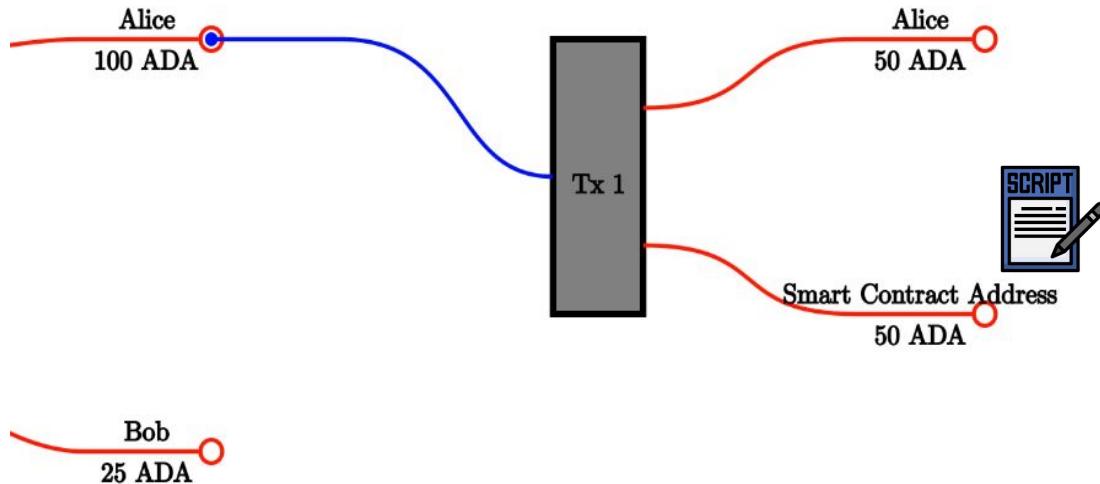
Alice
100 ADA

Alice quiere establecer **reglas** para controlar si sus monedas **se pueden gastar o no**. Así que escribe un **Script**.



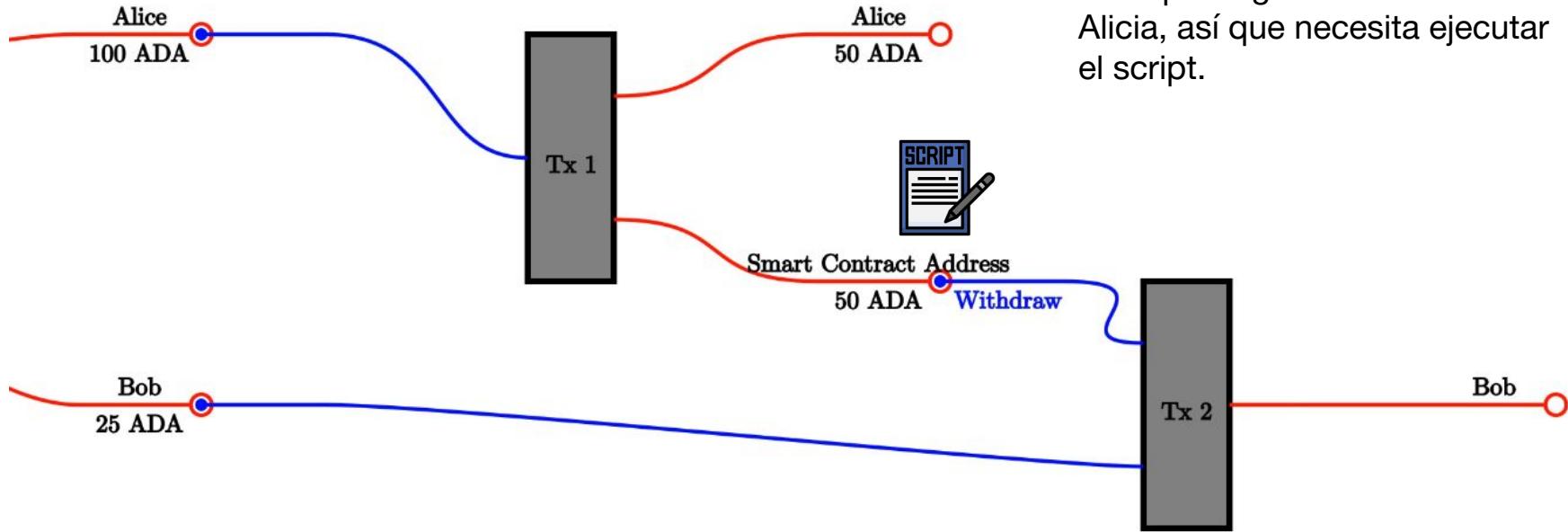
Bob
25 ADA

(E)UTxO: Contratos Inteligentes en acción



Alicia envía 50 ADAs a la dirección del Script y 50 de vuelta a su dirección.

(E)UTxO: Contratos Inteligentes en acción





Modelo (E)UTxO: Detalles de cada ítem

Modelo (E)UTxO: Detalles de cada ítem

Valor

χ : Value

Datum

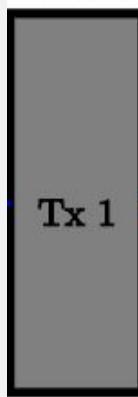
δ : Data

Redeemer

ρ : Redeemer

Contexto de
Transacción

σ : Tx



Tx 1

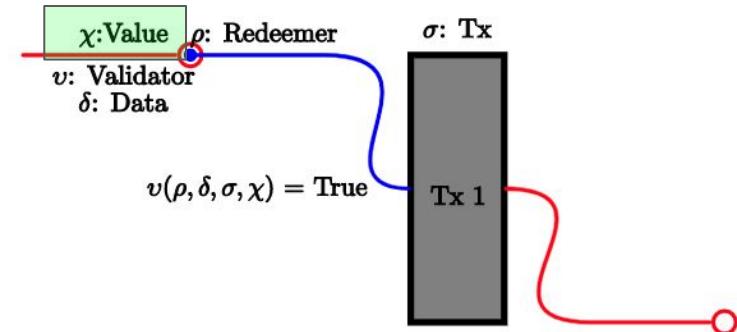
Validador/
Script

v : Validator
 $v(\rho, \delta, \sigma, \chi) = \text{True}$

Modelo (E)UTxO: Detalles de cada ítem - Valor

Que puede ser?:

- Lovelace (1 ADA = 1.000.000 Lovelace) ✓
- Lovelace + Tokens (FT y/o NFT) ✓
- Tokens sin Lovelace ✘



A tener en cuenta:

- **minUTxO** (Lovelace mínimos que tiene que tener el UTxO) depende del tamaño del UTxO.
- Se puede tener muchos Tokens al mismo tiempo, y se puede tener millones de un Token sin que haga la diferencia en costo o tamaño.

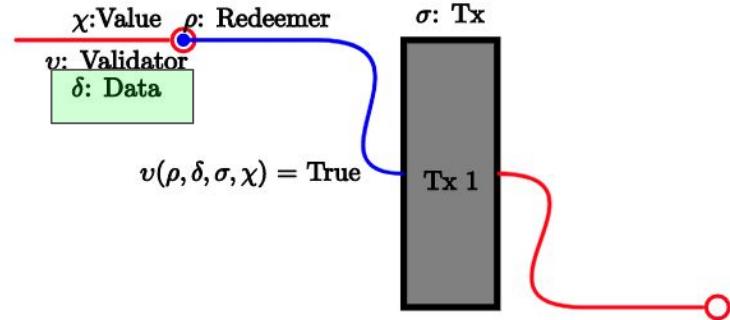
Modelo (E)UTxO: Detalles de cada ítem - Datum

Que puede ser?:

- Cualquier cosa (un número, texto, una lista de PKH, una estructura inventada, ...)

A tener en cuenta:

- El **creador del UTxO** tiene que comprometerse a un Datum al crear el UTxO. Pero no necesariamente proveerlo públicamente.



Creador Provee		Consumidor Provee
En UTxO	En Transacción	
Hash del Datum	-	Datum
Hash del Datum	Datum	Datum
Datum (inline)	-	-

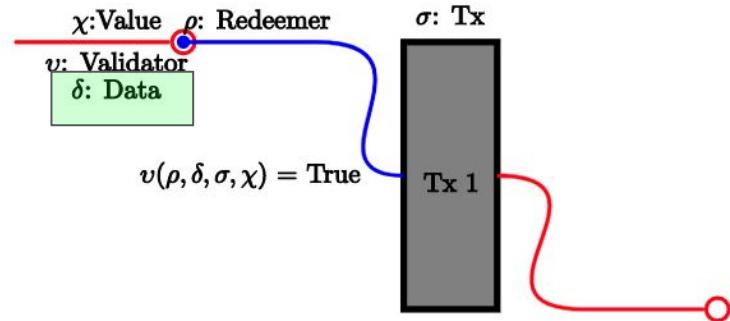
Modelo (E)UTxO: Detalles de cada ítem - Datum

Que puede ser?:

- Cualquier cosa (un número, texto, una lista de PKH, una estructura inventada, ...)

A tener en cuenta:

- El **creador del UTxO** tiene que comprometerse a un Datum al crear el UTxO. Pero no necesariamente proveerlo públicamente.



Creador Provee		Consumidor Provee
En UTxO	En Transacción	
Hash del Datum	-	Datum
Hash del Datum	Datum	Datum
Datum (inline)	-	-

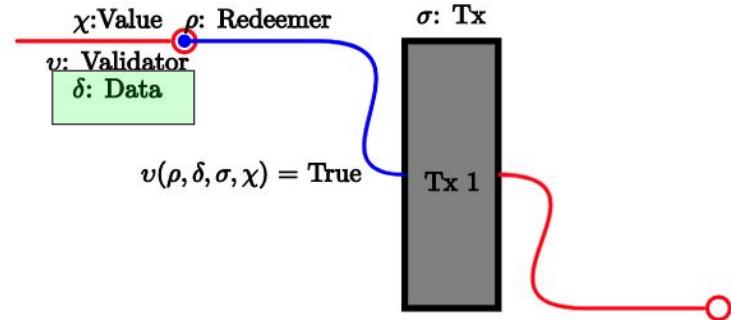
Modelo (E)UTxO: Detalles de cada ítem - Datum

Que puede ser?:

- Cualquier cosa (un número, texto, una lista de PKH, una estructura inventada, ...)

A tener en cuenta:

- El **creador del UTxO** tiene que comprometerse a un Datum al crear el UTxO. Pero no necesariamente proveerlo públicamente.



Creador Provee		Consumidor Provee
En UTxO	En Transacción	
Hash del Datum	-	Datum
Hash del Datum	Datum	Datum
Datum (inline)	-	-

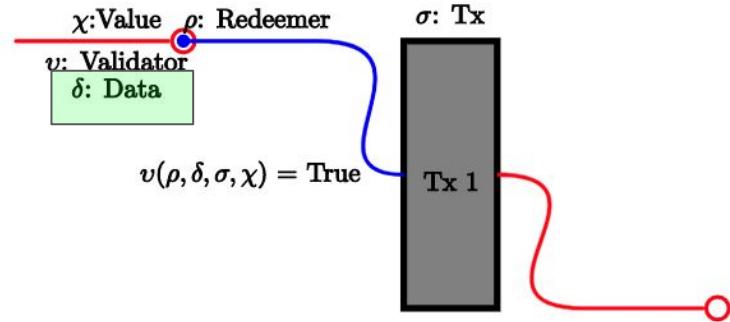
Modelo (E)UTxO: Detalles de cada ítem - Datum

Que puede ser?:

- Cualquier cosa (un número, texto, una lista de PKH, una estructura inventada, ...)

A tener en cuenta:

- El **creador del UTxO** tiene que comprometerse a un Datum al crear el UTxO. Pero no necesariamente proveerlo públicamente.



Creador Provee		Consumidor Provee
En UTxO	En Transacción	
Hash del Datum	-	Datum
Hash del Datum	Datum	Datum
Datum (inline)	-	-

Modelo (E)UTxO: Detalles de cada ítem - Redeemer

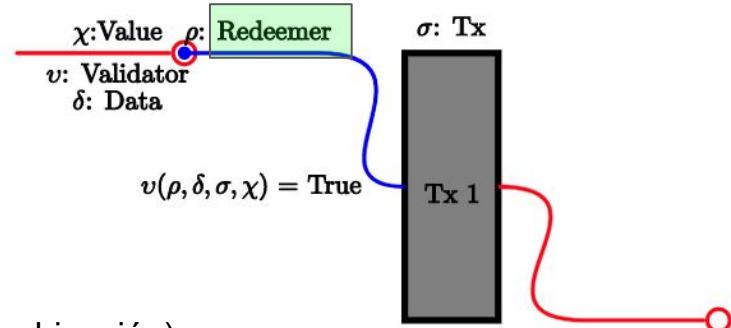
Que puede ser?:

- Cualquier cosa (un número, texto, una lista de PKH, una estructura inventada, ...)

Qué diferencia hay con el Datum?

- El Redeemer lo provee el que **consume** (**Quién va a gastar**) el UTxO

Usos **comunes** de Datums y Redeemers (en cualquier combinación):

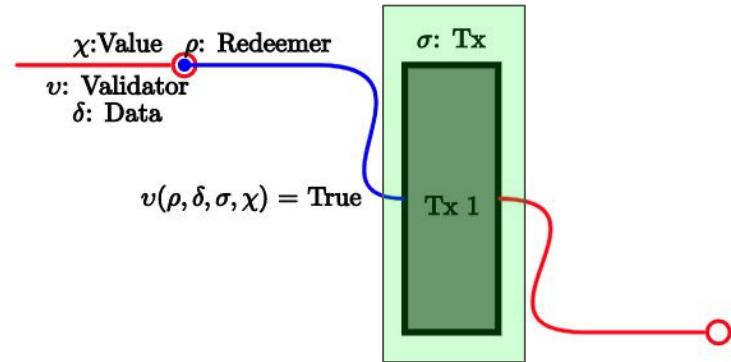


Datum	Redeemer
Quién/Cuándo/Con qué puedo consumir UTxO?	Razón de uso: Pedir prestamo, Pagar una cuota, Cancelar prestamo
Estado actual del UTxO	Información que sólo X persona sabe
Metadatos/"Configuraciones"	Valor por el cual reemplazar Datum

Modelo (E)UTxO: Detalles de cada ítem - Contexto de Transacción

A tener en cuenta:

- Sólo puede ser un valor de tipo **ScriptContext**.

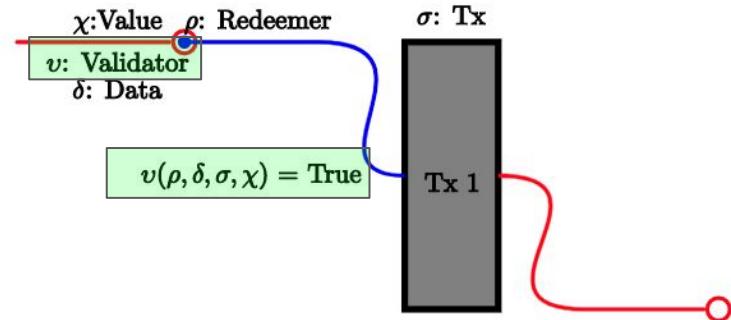


Qué contiene el valor **ScriptContext**:

- Tipo de Tx
- Rango temporal de validez
- Inputs (con sus respectivos Valores, Datums y Redeemers)
- Outputs (Con sus respectivos Valores y Datums)
- Referencias a UTxO y validadores relevantes pero que no son parte de la Tx
- Costo de la Tx
- Los que firmaron la Tx
- Identificador y cantidad de Tokens siendo acuñados/quemados en la Tx
- ...

Modelo (E)UTxO: Detalles de cada ítem - Validador/Script

- Tiene en cuenta todos los elementos que acabamos de ver para tomar la decisión si el UTxO puede ser consumido por la Tx.
- Si la Tx consume múltiples UTxO, cada UTxO tiene su propio validador que decide sólo por ese UTxO.
- Para que la Tx sea exitosa, los validadores de todos los UTxO tienen que devolver **True**.
- Hay 6 tipos de validadores. En este workshop vamos a aprender a usar los 2 más comunes:
 - **Spend:** El que se ejecuta para permitir/denegar consumir un UTxO.
 - **Mint:** El que se ejecuta para acuñar o quemar un token.



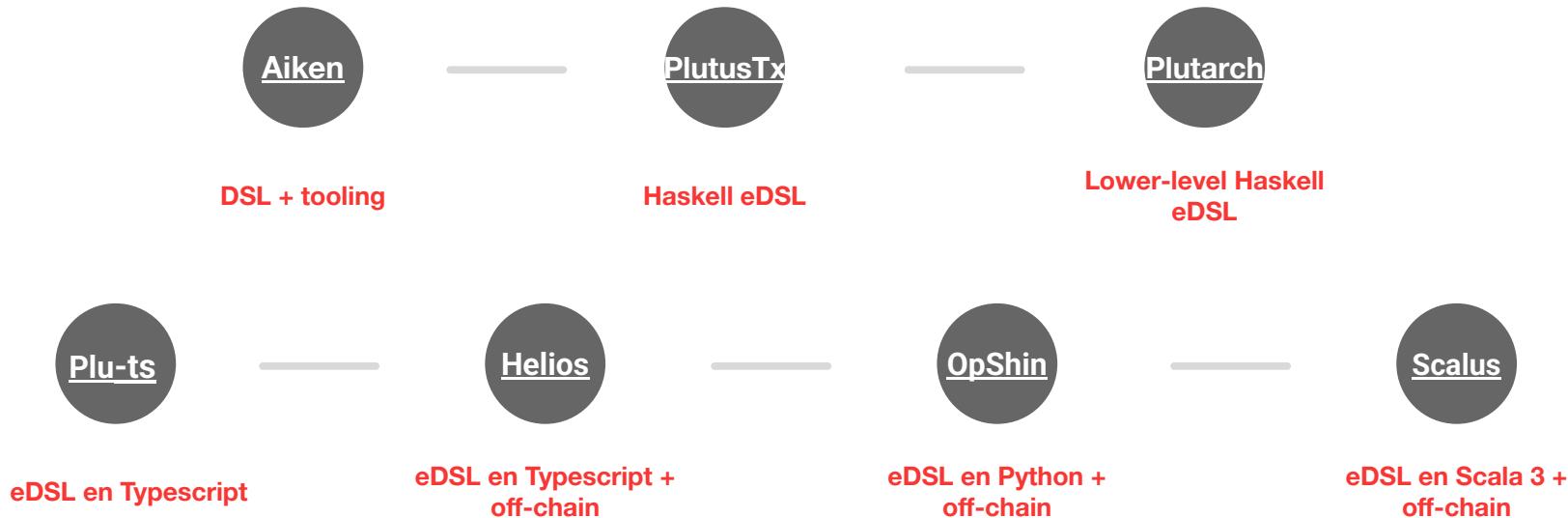
(E)UTxO: ¿Qué tipo de reglas podemos establecer?

- Quién puede gastar los activos
- Cuándo se pueden gastar los activos
- ¿Necesitamos crear/o acuñar tokens?
- ¿Necesito gastar tokens?
- Entre otros
- Mezcla de uno o más



Smart Contracts

Herramientas - Lenguajes para escribir contratos inteligentes en Cardano



+ Introducción a Aiken

44

AIKEN



¿Qué es Aiken?

Es un lenguaje **de programación y un conjunto de herramientas para desarrollar contratos inteligentes** en la blockchain de Cardano.

Está orientado a la robustez y a la experiencia del desarrollador. Inspirado en lenguajes como Gleam, Rust y Elm.

Para leer más al respecto, visite el sitio web en: <https://aiken-lang.org/>

Herramientas - ¿Por qué Aiken?



El 79.3% adopta Aiken para escribir validadores.



Fácil y seguro

El objetivo es permitirte comenzar el desarrollo de contratos inteligentes rápidamente, con la confianza de que tu código on-chain está funcionando correctamente.



Excelente experiencia para los desarrolladores

Ofrece integraciones con editores, mensajes de error amigables, retroalimentación rápida de pruebas y fácil generación de documentación.



Configuración inicial mínima

Busca minimizar la configuración inicial, proporcionando funcionalidad lista para usar.



Diseño modular

Su arquitectura tiene en cuenta la modularidad, permitiendo a los desarrolladores elegir y seleccionar componentes según sea necesario, fomentando la interoperabilidad y el crecimiento de la comunidad.

Características

Aiken: Propiedades clave del lenguaje

- DSL para validadores en Cardano (Nodos de Cardano son el único target)
- Programación declarativa funcional ( POO)
- Funciones de primera clase (todo es una expresión)
- Inmutabilidad
- Tipado estático con inferencia y genéricos
- Tipo de datos algebraico (ADTs)
- Recursión
- Puro
- Sistema de módulos

Herramientas - Aiken - Herramientas

La plataforma de Aiken combina el compilador principal con un conjunto de herramientas, documentación, bibliotecas y recursos que facilitan el desarrollo, verificación, validación e implementación de contratos inteligentes en la blockchain de Cardano.

 stdlib Librería Estándar Funciones, tipos, constantes y alias utilizados en la mayoría de los casos.	 aikup Gestor de herramientas CLI para gestionar múltiples versiones de Aiken.
 aiken LSP Servidor LSP para aiken.	 VSCode Vim / Neovim Emacs Integraciones con editores Plugins que proporcionan resaltado de sintaxis y reglas de indentación para Aiken.
 play Aiken Playground Playground para probar Aiken sin instalarlo	 awesome-aiken Recursos Colección de bibliotecas, DApps, tutoriales y otras cosas relacionadas con Aiken.

CLI todo en uno:

- **Proyecto:** Crear, formatear, verificar tipos y compilar proyectos. Gestionar dependencias. Generar documentación y planos de validadores.
- **Validador:** Calcular la dirección del validador, ejecutar pruebas (pruebas integradas). Simular la transmisión y trabajar con UPLC.

Ejercicio - Aiken - Instalación

<https://github.com/ohkedu/cdw-azteca2025>



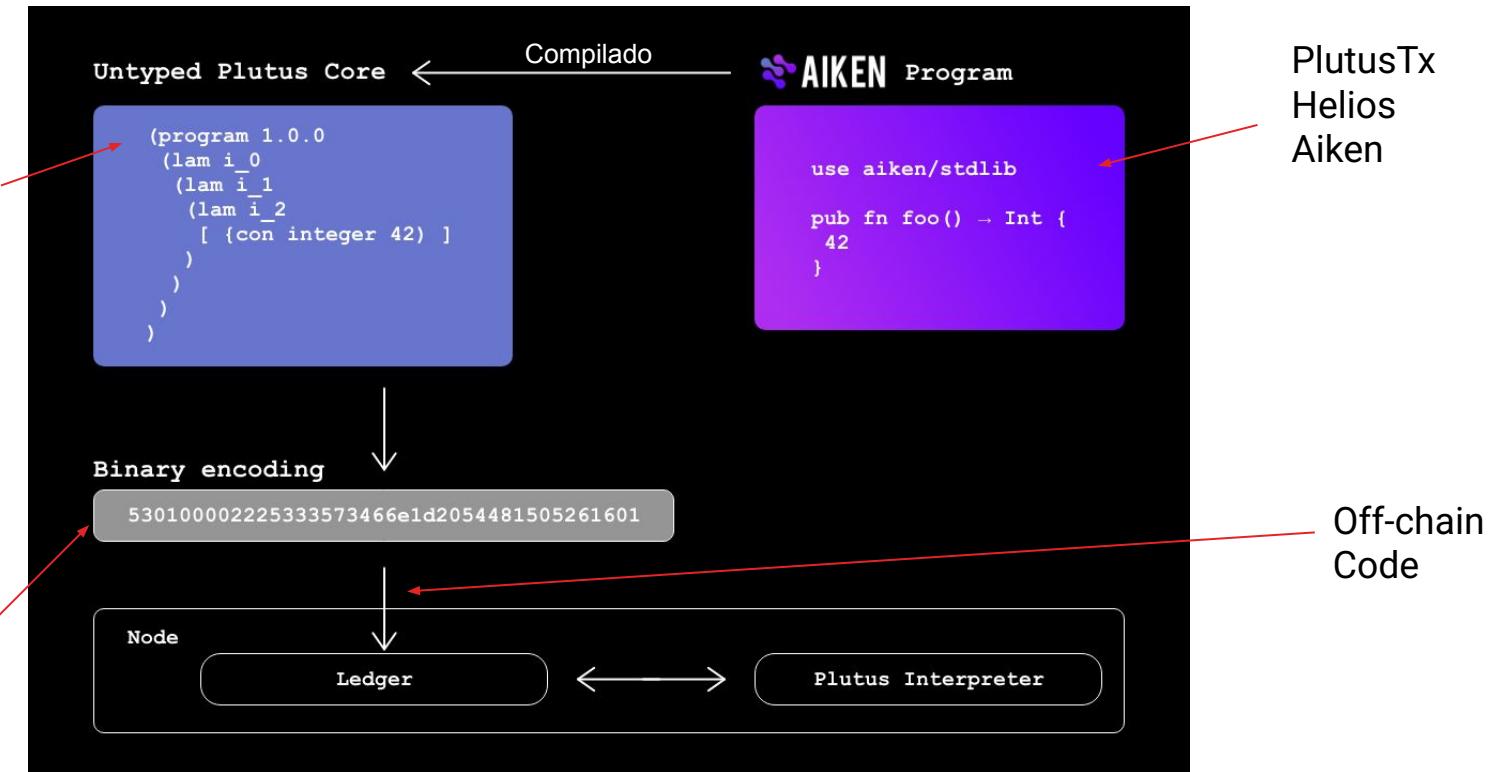
¿Qué significa on-chain y off-chain?

Roles del código on-chain y off-chain en un contrato inteligente

- **On-chain:** Código que se ejecuta en el nodo durante la adición de nuevas transacciones e información que se almacena en la blockchain.
- **Off-chain:** Código que se ejecuta en el dispositivo del usuario (o proveedor de servicios) para consultar la blockchain y construir y enviar transacciones.

Compile on-chain code

UPLC es el lenguaje que los **nodos** saben **interpretar** y, por lo tanto, el que se **ejecuta** en la blockchain.



UPLC se codifica a binario antes de enviarse.

Ejercicio - Aiken - Estructura general

Ejercicio - Aiken - Always true

Always True:

1. Crear wallet y obtener ADAs desde el faucet o pedir a los instructores
2. Obtener una API_KEY desde Blockfrost - TODO: tener 3 API's key y poner los pasos a seguir para obtenerla.
3. Crear un validador que permita siempre consumir el UTxO.
4. Copiar el código del validador desde el archivo plutus.json al simulador de MeshJS.
5. Enviar un UTxO al validador utilizando un datum válido.
6. Consumir el UTxO.



Ejercicio - Aiken - Signature

Signature (En pareja):

1. Crear un validador que permita consumir el UTxO sólamente si la transacción es firmada por la persona definida en el Datum del UTxO.
2. Enviar un UTxO definiendo como datum el Hash de su compañero.
3. Tratar de consumir el UTxO que le asigne a mi compañero para probar que no puede ser consumido (Opcional).
4. Consumir el UTxO que me fue asignado.



Aiken: Vesting Validator

Un **validador** de "vesting" es un contrato inteligente que **controla la liberación** de tokens basado en una **condición de tiempo específica**. Cuando se utiliza un validador de "vesting" con una sola liberación:

- Todos los tokens permanecen bloqueados hasta que un período de tiempo predeterminado, conocido como el período de "vesting", ha pasado.
- Una vez que el período de "vesting" termina, la cantidad total de tokens se libera de una vez. Este mecanismo asegura que el destinatario solo pueda acceder a los tokens después de cumplir con el compromiso de tiempo requerido.



Modelo (E)UTxO: Ejemplo Vesting - Diseño

1. En el momento de que el empleado acepta la posición, el empleador va a generar **4 UTxO con 25% de las acciones en cada uno.**
2. Los 4 UTxO **chequean** que el que quiere consumir el UTxO **sea el empleado y que una fecha límite haya pasado.**
3. Cada UTxO tiene una **fecha límite distinta** (Ej. 1/1/2025, 1/1/2026, 1/1/2027, 1/1/2028)
4. El beneficiario sólo tiene que **esperar** que pase la fecha límite de cada uno **para poder consumir el UTxO** y obtener sus acciones.

Modelo (E)UTxO: Ejemplo Vesting - Idea

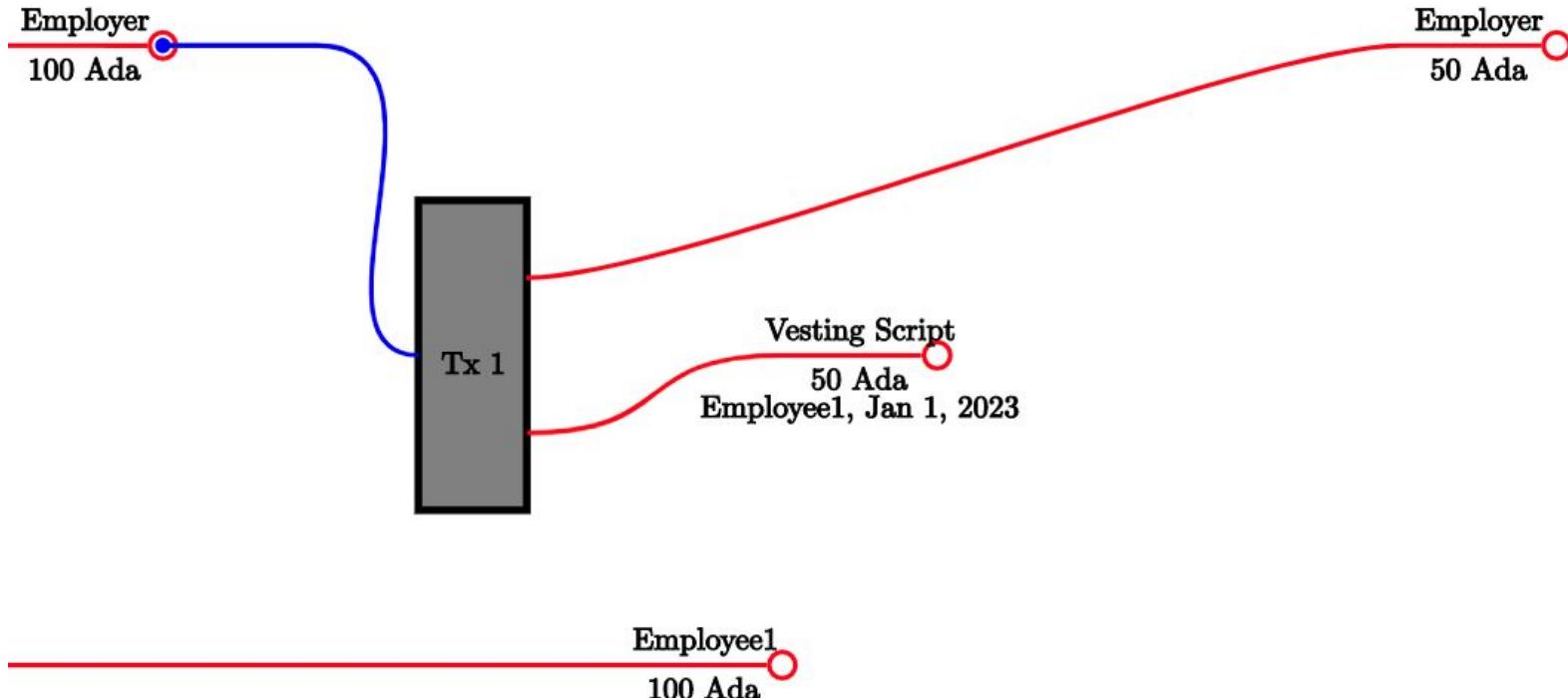
- Muchas compañías tipo Facebook, Google, Apple, y Netflix, proveen “**Restricted Stocks Units**” (RSU) a los empleados como parte de la compensación.
- Las “Unidades de acciones restringidas” tienen usualmente un calendario de por ejemplo **25% de las acciones por año durante 4 años**.
- Entre otras razones, esto se hace para que el empleado:
 - No venda el 100% de las acciones a la vez (potencialmente impactando el precio del mercado)
 - No las venda y se vaya al otro día.

Modelo (E)UTxO: Ejemplo Vesting - Visualización

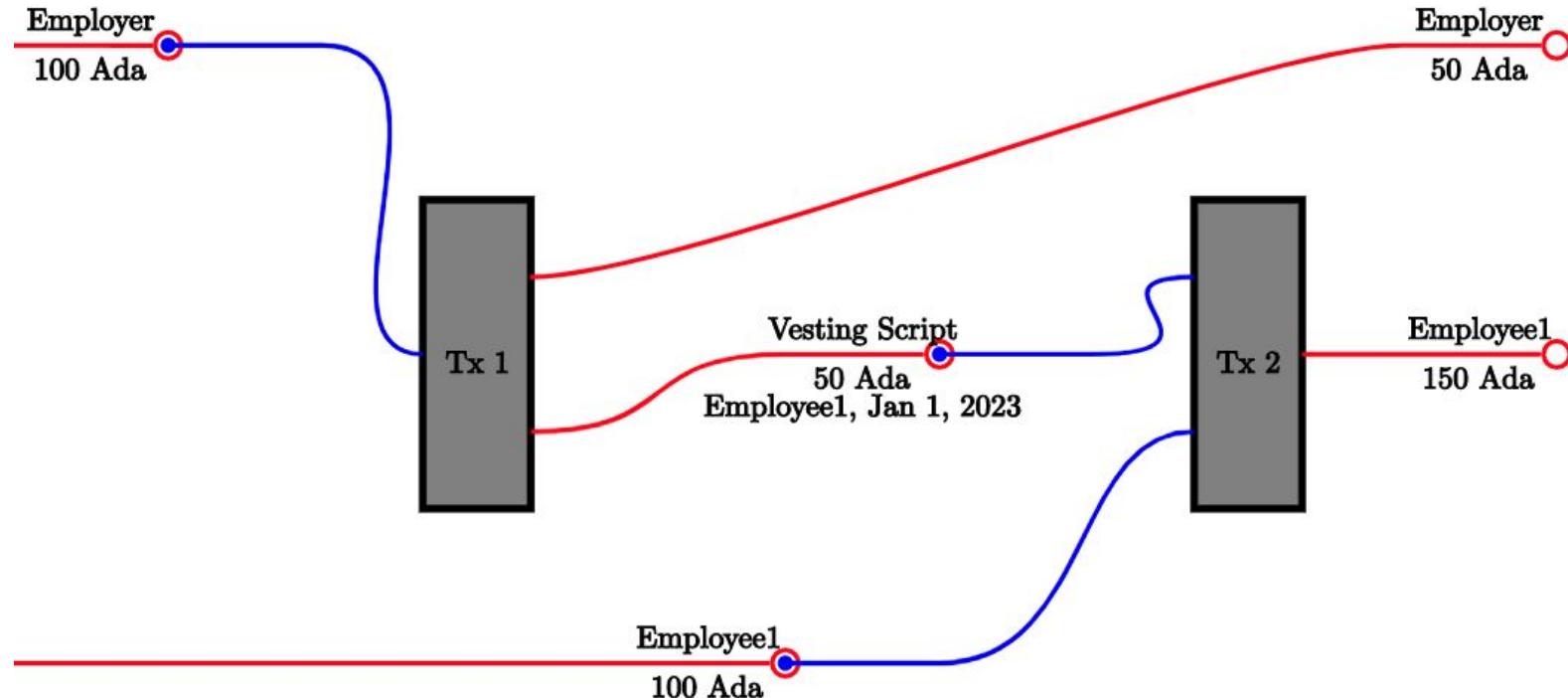
Employer
100 Ada

Employee1
100 Ada

Modelo (E)UTxO: Ejemplo Vesting - Visualización



Modelo (E)UTxO: Ejemplo Vesting - Visualización



Ejercicio - Aiken - Vesting

Vesting (En pareja):

1. Crear un validador que permita consumir el UTxO sólamente si la transacción es firmada por la persona definida en el Datum del UTxO y además es después de una hora definida.
2. Enviar un UTxO definiendo como Datum el Hash de mi compañero y el tiempo.
3. Tratar de consumir el UTxO que le asigne a mi compañero con una billetera diferente o antes del tiempo acordado para probar que no puede ser consumido. (Opcional)
4. Consumir el UTxO que me fue asignado y después del tiempo acordado.

Gracias !

Preguntas?



INPUT | OUTPUT

IOG: Educación

- **Youtube Channel:** IOG Academy
- **Plutus Pioneer Program:**
[input-output/plutus-pioneer-program](https://github.com/input-output-plutus/pioneer-program)
on github
- **Cardano Documentation:**
docs.cardano.org/
- Community, discord server.

