¨INSTITUTO SUPERIOR TECNOLÓGICO PRIVADO VALLE GRANDE¨

CARRERA PROFESIONAL DE ANÁLISIS DE SISTEMAS

VALLE GRANDE
INSTITUTO DE EDUCACIÓN SUPERIOR TECNOLÓGICO PRIVADO

📑 Manual de Ejecución — Préstamo con LoanManager

Paso 0 — Verificar estado inicial (sanity checks)

- Usar **Cualquier cuenta** en Remix → llamar `MockERC20.balanceOf(<address>)`:
    - `balanceOf(Cuenta A)` debe ser `0`
    - `balanceOf(Cuenta B)` debe ser `0`
- Llamar `LoanManager.nextLoanId()` → debe devolver `1`
- Llamar `LoanManager.loans(1)` → debe estar vacío / no funded

Por qué: confirmar que no hay fondos/token ni préstamos previos que contaminen la prueba.

Paso 1 — Solicitar préstamo (Borrower)

Usar: Cuenta A
Contrato / Función:

```
LoanManager.requestLoan(uint256 principal, uint256 interestBP, uint256 durationSeconds)
```

Inputs (copiar a Remix):

- `principal = 1000000000000000000` (=> **1 token**, 1e18 wei)
- `interestBP = 500` (=> **500 basis points = 5%**)
- `durationSeconds = 86400` (=> 1 día)
- **Value (ETH)** en el campo `value` de Remix = `2000000000000000000` (=> **2 ETH** colateral)
- Verificar evento `LoanRequested` en los logs → `loanId = 1`

Por qué: Esto crea la solicitud de préstamo y deposita el colateral de 2 ETH en el LoanManager.

Paso 2 — Mintear tokens para el prestamista (Lender)

Usar: Cuenta B
Contrato / Función:

```
MockERC20.mint(address to, uint256 amount)
```

Inputs:

- `to = 0x4631fB354D128dcf0c5cdb6526F9B6712efec1b4` (Cuenta B)
- `amount = 2000000000000000000` (=> **2 tokens**)
- Verificar `MockERC20.balanceOf(Cuenta B)` ≥ `1000000000000000000` (≥ 1 token)

Por qué: das liquidez al prestamista para poder financiar el préstamo.

Paso 3 — Aprobar transferencias al LoanManager (Lender)

Usar: Cuenta B
Contrato / Función:

`MockERC20.approve(address spender, uint256 amount)`

Inputs:

- `spender = <LoanManager address>`
- `amount = 2000000000000000000` (=> 2 tokens)
- Verificar `MockERC20.allowance(Cuenta B, LoanManager)` ≥ `1000000000000000000` (≥ 1 token)

Por qué: permite al LoanManager mover tokens desde la cuenta del prestamista cuando financie el loan.

Paso 4 — Financiar préstamo (Lender)

Usar: Cuenta B
Contrato / Función:

`LoanManager.fundLoan(uint256 loanId)`

Inputs:

- `loanId = 1`
- `Value = 0`
- Verificar evento `LoanFunded`
- `LoanManager.loans(1).funded == true`

Por qué: el prestamista compromete los tokens al préstamo para que el borrower los pueda retirar luego.

Paso 5 — Retirar tokens (Borrower)

Usar: Cuenta A
Contrato / Función:

`LoanManager.withdrawLoan(uint256 loanId)`

Inputs:

- `loanId = 1`
- Verificar evento `LoanWithdrawn`
- `MockERC20.balanceOf(Cuenta A) == 1000000000000000000` (=> **1 token**)

Por qué: el borrower retira el principal (1 token) que financió el lender.

Paso 6 — Aprobar repago (Borrower)

Usar: Cuenta A
Contrato / Función:

`MockERC20.approve(address spender, uint256 amount)`

Inputs:

- `spender = <LoanManager address>`
- `amount = 1050000000000000000` (=> **1.05 tokens**: principal + 5% interés)
- Verificar `MockERC20.allowance(Cuenta A, LoanManager)` ≥ `1050000000000000000`

Por qué: autorizas al LoanManager a cobrar el principal + interés para completar el repago.

Paso 7 — Pagar préstamo (Borrower)

Usar: Cuenta A
Contrato / Función:

`LoanManager.repayLoan(uint256 loanId)`

Inputs:

- `loanId = 1`
- Verificar evento `LoanRepaid`
- `LoanManager.loans(1).repaid == true`
- `MockERC20.balanceOf(Cuenta B)` ≈ `1050000000000000000` (=> **~1.05 tokens**)

Por qué: se transfiere del borrower al lender el principal + interés; cierra el préstamo.

Paso 8 — Recuperar colateral (Borrower)

Usar: Cuenta A
Contrato / Función:

`LoanManager.claimCollateral(uint256 loanId)`

Inputs:

- `loanId = 1`
- Verificar que **Cuenta A** recibe `2000000000000000000` wei (=> **2 ETH**) de vuelta

Por qué: una vez repagado, el borrower reclama su colateral.

Resultado final — Verificaciones rápidas

- `MockERC20.balanceOf(Cuenta A)` final: **0 tokens** (pagó 1.05; tenía 1)
- Ether de **Cuenta A**: +2 ETH (recuperó colateral)
- `MockERC20.balanceOf(Cuenta B)` final: **~1.05 tokens** (ganancia 0.05)
- `LoanManager.loans(1).repaid == true` y `funded == true` (cerrado)