Stake function allows staking NFTs with token ID 0

```
function stake(GalileoStakingStorage.StakeTokens calldata stakeTokens) external
whenNotPaused nonReentrant {
    // Recover and verify the voucher signature to ensure its authenticity.
    _recover(stakeTokens);
    // Call the internal function to handle the actual staking process
    _stakeTokens(
      stakeTokens.collectionAddress,
      stakeTokens.tokenId,
      stakeTokens.citizen,
      stakeTokens.timelockEndTime,
      stakeTokens.stakedLeox
    );
  }
```

Root Cause

- `stake()` → calls `_stakeTokens()` without **any restriction on** `tokenId`.
- `_stakeTokens()` accepts `tokenId == 0`.
- But later, in `unstake()` (or related functions), the logic **blocks** `tokenId == 0` (either by explicit check or by assuming IDs > 0).

This creates an **inconsistent state machine**:

- Stake with `tokenId == 0` succeeds.
- Unstake with `tokenId == 0` is impossible.

Attack / Edge Case Scenario

1. Alice owns an NFT with `tokenId = 0`.
2. She stakes it via `stake()`.
    - ✅ `_stakeTokens()` accepts it.
3. Later, she calls `unstake()` for her NFT.
    - ❌ Fails, because `unstake()` disallows `tokenId == 0`.
4. Result: Alice's NFT is **permanently locked in the contract**.

Impact

- **User funds (NFTs) locked** irretrievably.
- **DoS for tokenId=0 holders** (they cannot exit).
- Breaks **trust and usability**: contract does not behave consistently.
- Depending on project size, could become a **major UX bug** or even **loss of user assets** if not fixable.

Fix Recommendations

Two ways to solve this cleanly:

1. **Disallow staking of `tokenId == 0` (safe option)**
   - Add validation in `stake()` or `_stakeTokens()`:
     ```
     require(tokenId != 0, "Invalid tokenId");
     ```
   - Prevents users from ever locking un-withdrawable NFTs.
2. **Support unstaking of `tokenId == 0` (better UX)**
   - Modify `unstake()` logic to handle `tokenId == 0` consistently.
   - Ensures symmetry: “if you can stake it, you can unstake it.”
3. **Invariant rule enforcement (best practice)**
   - Whatever is allowed in `stake()` must be allowed in `unstake()`.
   - Otherwise you create a one-way lock bug.