```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract IncidentReport {
    // Estructura para los reportes
    struct Report {
        uint id;            // Identificador único
        string incidentType; // Tipo de incidente (robo, asalto, etc.)
        string location;    // Ubicación aproximada
        uint timestamp;     // Momento del reporte
        address reporter;   // Dirección del usuario que reportó
    }

    // Mapeo para almacenar los reportes
    mapping(uint => Report) public reports;
    uint public reportCount = 0;

    // Evento para notificar cuando se agrega un nuevo reporte
    event ReportAdded(uint id, string incidentType, string location, uint
timestamp, address reporter);

    // Función para registrar un reporte
    function addReport(string memory _incidentType, string memory _location)
public {
        reportCount++;
        reports[reportCount] = Report(reportCount, _incidentType, _location,
block.timestamp, msg.sender);
        emit ReportAdded(reportCount, _incidentType, _location,
block.timestamp, msg.sender);
    }
}
```

este contrato guarda un registro básico de incidentes en la blockchain.

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HeatMap {
    // Mapeo para contar incidentes por área
    mapping(string => uint) public areaReports;

    // Evento para registrar un nuevo reporte en un área
    event ReportLogged(string area, uint count);

    // Función para agregar un reporte a un área
    function logReport(string memory _area) public {
        areaReports[_area]++;
        emit ReportLogged(_area, areaReports[_area]);
    }

    // Función para consultar cuántos reportes tiene un área
    function getReports(string memory _area) public view returns (uint) {
        return areaReports[_area];
    }
}
```

Este contrato cuenta el número de reportes en cada área (por ejemplo, un código postal).

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract RewardSystem {
    address public admin;
    mapping(address => uint) public rewards; // Recompensas por usuario

    // Evento para notificar una recompensa
    event RewardGiven(address user, uint amount);

    constructor() {
        admin = msg.sender; // El creador del contrato es el administrador
    }

    // Función para dar recompensas (solo admin puede usarla)
    function giveReward(address _user, uint _amount) public {
        require(msg.sender == admin, "Only admin can give rewards");
        rewards[_user] += _amount;
        emit RewardGiven(_user, _amount);
    }

    // Función para que los usuarios consulten su saldo de recompensas
    function getRewards(address _user) public view returns (uint) {
        return rewards[_user];
    }
}
```

Este contrato recompensa a los usuarios con tokens por reportes que sean validados.