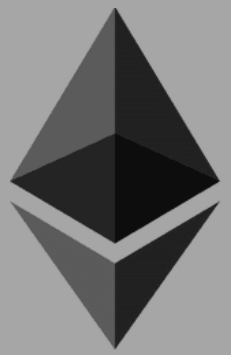
ESME sudria
PARIS | BORDEAUX | LILLE | LYON

Blockchain

Présenté par Lucas Gobbi, Lisa Journo & Joffrey Chavanette
Projet proposé par M. Ozcan



ethereum

PROJECT TEAM

Lucas Gobbi

Lisa Journo

Joffrey Chavanette

PLAN

- ❑ Project Presentation
- ❑ Main Features
- ❑ Benefits & Drawbacks
- ❑ UML Diagram
- ❑ Smart Contract

PROJECT PRESENTATION

NICE TO YOU

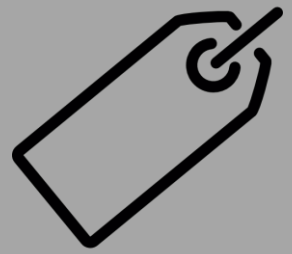
Farmer

Label

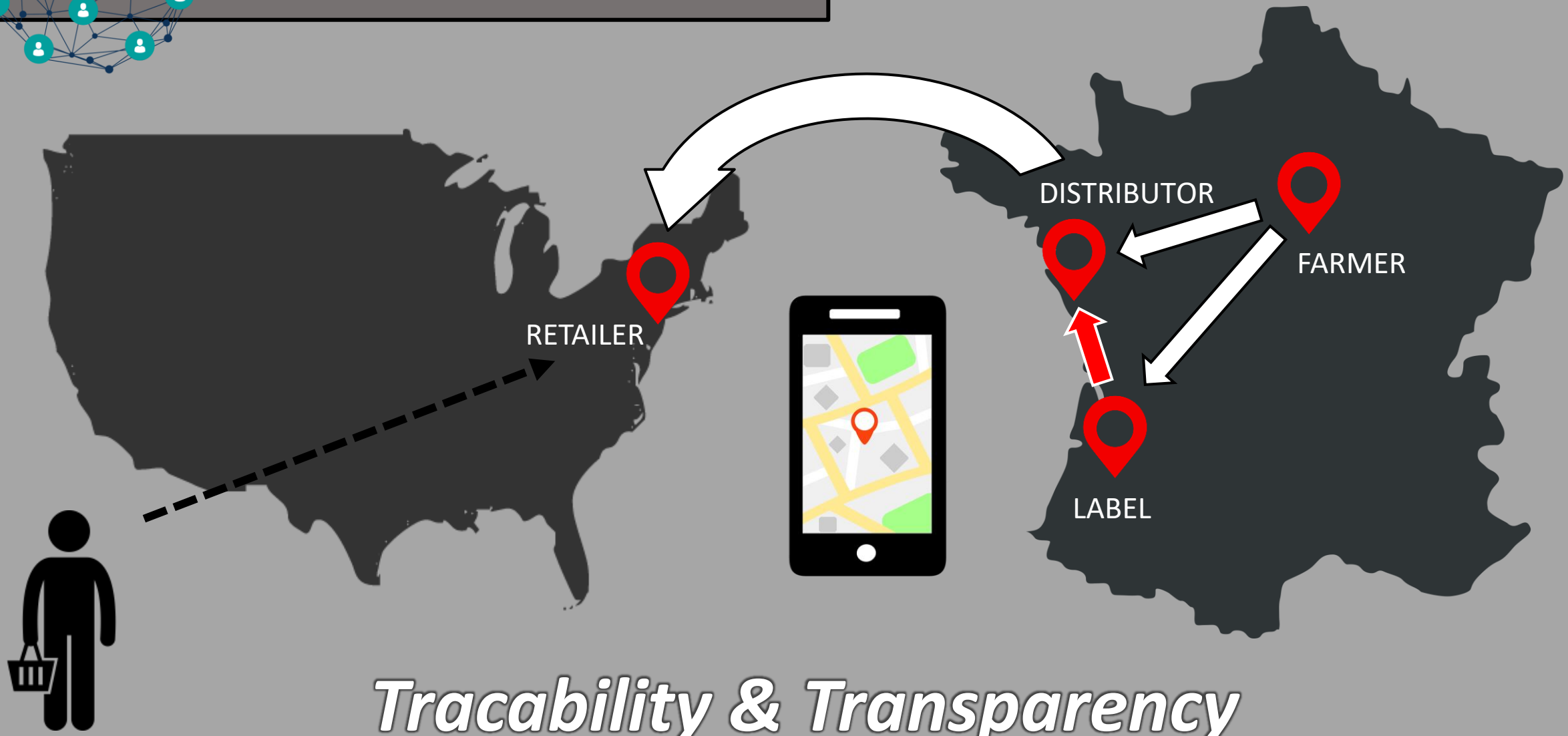
Distributor

Retailer

Consumer



MAIN FEATURES



Tracability & Transparency

Benefits & Drawbacks

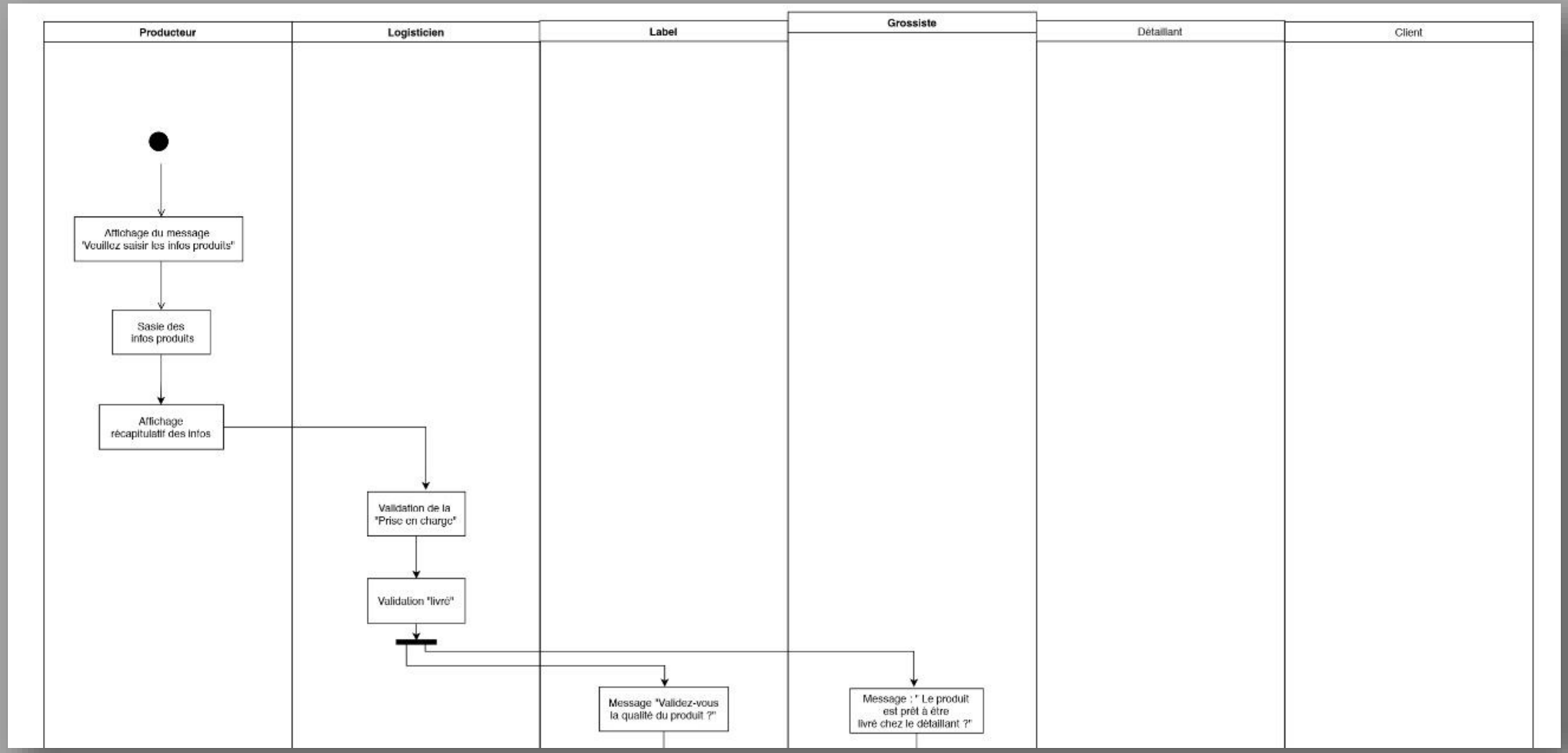
Benefits

- ✓ Tracability
- ✓ Transparency
- ✓ Security
- ✓ Productivity Gain

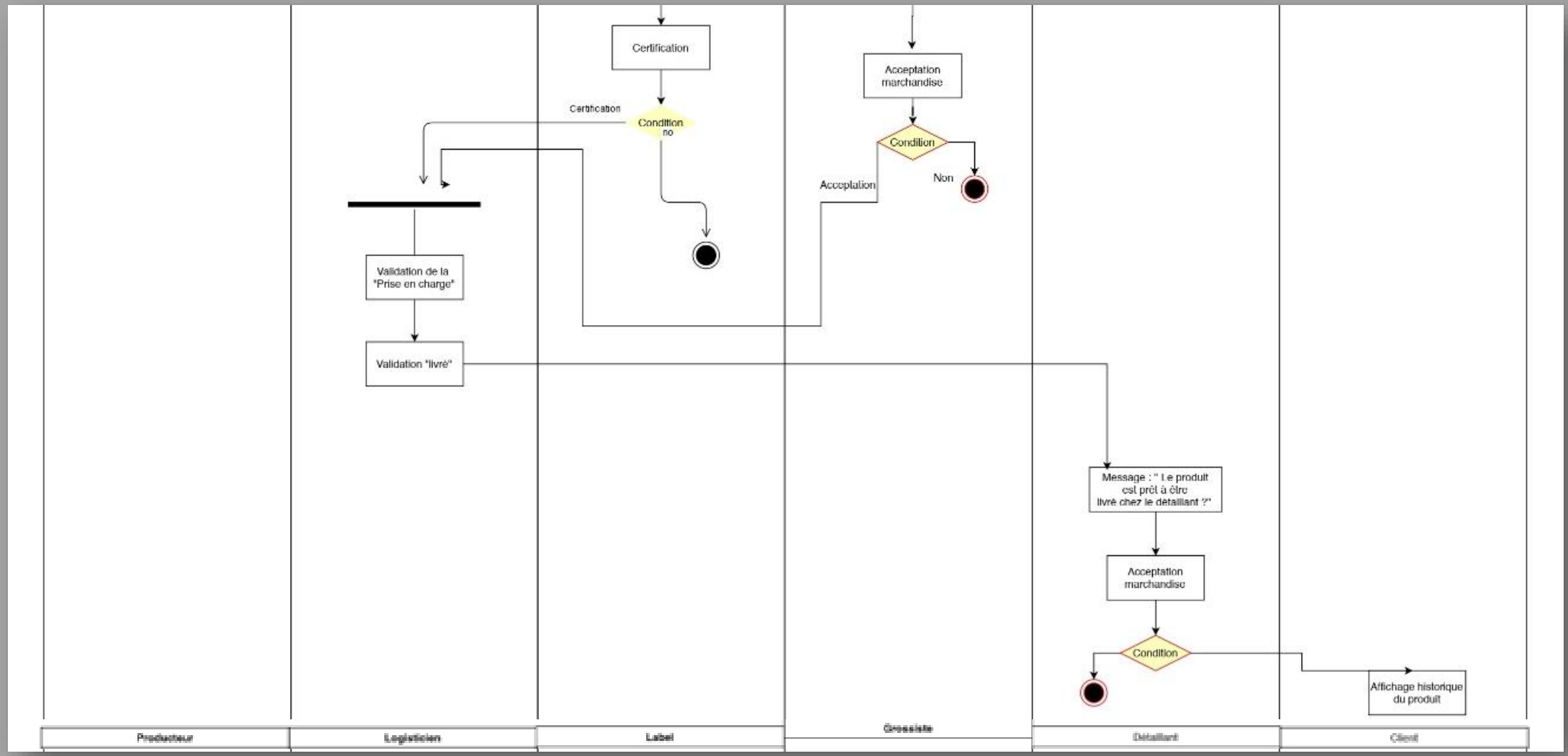
Drawbacks

- ✓ Deployment costs
- ✓ Transaction fees
- ✓ Trust issue

UML DIAGRAM (1)



UML DIAGRAM (2)



SMART CONTRACT

```
// Déclaration de la structure 'Item' avec les champs suivant:

struct Item {
  uint    meatID;                 // Numéro de lot assigné par le producteur, équivalent à un code bar qui peut être vérifier par le client
  address ownerID;                // Adressse Metamask-Ethereum du propriétaire du produit à un instant t
  address originFarmerID;         // Adresse Metamask-Ethereum du producteur
  string  originFarmName;         // Farmer Name
  string  originFarmInformation;  // Orifine de la viande (région)
  string  originMeatType;
  uint256 productDate;            // Product Date NOTE: MIGHT NEED TO CHANGE type
  State   itemState;              // Avancement du produit dans la chaine de productin et distribution
  Lab     itemLab;
  address distributorID;          // Metamask-Ethereum address of the Distributor
  address labelID;
  address retailerID;             // Metamask-Ethereum address of the Retailer
}

// Block number stuct

struct Txblocks {
  uint FTD; // blockfarmerToDistributor
  uint DTR; // blockDistributorToRetailer
}


event ProduceByFarmer(uint meatID);         //0
event ShippedByFarmer(uint meatID);         //1
event ReceivedByDistributor(uint meatID);   //2
event ReceivedByLabel(uint meatID);         //I1
event ValidatedByLabel(uint meatID);        //I2
event ShippedByDistributor(uint meatID);    //3
event ReceivedByRetailer(uint meatID);      //4
```

- ✓ Structure
- ✓ Etapes
- ✓ Rendu

SMART CONTRACT

ESME sudria
PARIS | BORDEAUX | LILLE | LYON

```
// Définit un contrat nommé "FarmerRole" qui gere le role du meme nom (possibilité d'ajout, suppression et vérification)
contract FarmerRole {
  using Roles for Roles.Role;

  // Creation de 2 évènements : Ajout et Suppression.
  event FarmerAdded(address indexed account);
  event FarmerRemoved(address indexed account);

  // Definition d'une structure 'farmers' héritée de la librairie 'Roles'.
  Roles.Role private farmers;

  // Par défaut, cette fonction ajoute le propriétaire du contrat comme 1er producteur
  constructor() public {
    _addFarmer(msg.sender);
  }

  // Définit un modifier qui vérifie si l'envoyeur de la commande a le role approprié.
  modifier onlyFarmer() {
    require(EstProducteur(msg.sender));
    _;
  }

  // Definit une fonction 'EstProducteur' pour vérifier si l'adresse a ce role
  // Si l'adresse a ce role, la fonction booleenne retourne 'True'
  function EstProducteur(address account) public view returns (bool) {
    return farmers.has(account);
  }

  // Définit une fonction d'"AjoutProducteur" pour ajouter une adresse comme producteur
  function AjoutProducteur(address account) public onlyFarmer {
    _addFarmer(account); // Ajoute le role de producteur a cette adresse
  }

  // Définit une fonction de "SuppProducteur" pour supprimer une adresse du role de producteur
  function SuppProducteur() public {
    _removeFarmer(msg.sender);
  }
```

```
function add(Role storage role, address account) internal {
  require(account != address(0));
  require(!has(role, account));

  role.admin[account] = true;
}

/**
 * @dev enleve le droit d'acces a ce role sur un contrat suivant l'adresse
 */
function remove(Role storage role, address account) internal {
  require(account != address(0));
  require(has(role, account));

  role.admin[account] = false;
}

/**
 * @dev verifie que le compte (l'adresse) a bien un droit d'acces
 * @return bool
 */
function has(Role storage role, address account)
  internal
  view
  returns (bool)
{
  require(account != address(0));
  return role.admin[account];
}
```

- ✓ Structure
- ✓ Fonctions
- ✓ Rendu

SMART CONTRACT

```
modifier onlyOwner() {
    require(EstProprietaire());
    _;
}

// Fonction permettant de vérifier que l'adresse appelé appartiene bien à la bonne personne

function EstProprietaire() public view returns (bool) {
    return msg.sender == origOwner;
}

// Définition de la fonction permettant de retirer les droits du propriétaire du contrat

function SuppProprieter() public onlyOwner {
    emit TransferOwnership(origOwner, address(0));
    origOwner = address(0);
}

// Définition d'une fonction publique permettant le changement de propriétaire (tout le monde peut attribuer des droits)

function ChangementPropriter(address newOwner) public onlyOwner {
    _transferOwnership(newOwner);
}

// Définition d'une fonction locale permettant le changement de propriétaire (seulement le propriétaire peut attribuer des droits)

function _transferOwnership(address newOwner) internal {
    require(newOwner != address(0));
    emit TransferOwnership(origOwner, newOwner);
    origOwner = newOwner;
}
```

- ✓ Structure
- ✓ Fonctions
- ✓ Rendu