**Приложение A**

|  |
| --- |
| struct Organization {  String name  PublicKey publicKey  Integer maxPeopleCapacity  String advertisementAddress  Map<Integer, Integer> visits  }  struct MedicalEmployee {  String name  PublicKey publicKey  }  struct Admin {  String name  PublicKey publicKey  }  contract AccessController {  Map<PublicKey, Organization> organizations  Map<PublicKey, MedicalEmployee> medicalEmployees  Map<PublicKey, Admin> admins  addOrganization(name, publicKey, maxPeopleCapacity, advertisementAddress){  if actorRole != 'admin' **revert**  if organizations.contains(publicKey) **revert**  organizations[publicKey] = new Organization(  name,  publicKey,  maxPeopleCapacity,  advertisementAddress  )  }  addMedicalEmployee(name, publicKey){  if actorRole != 'admin' return  if medicalEmployees.contains(publicKey) return  medicalEmployees[publicKey] = new MedicalEmployee(  name,  publicKey,  )  }  addAdmin(name, publicKey){  if actorRole != 'superadmin' **revert**  if admins.contains(publicKey) **revert**  admins[publicKey] = Admin(  name,  publicKey,  )  }    Organization getOrganizationInfo(publicKey) {  return organizations[publicKey]  }  } |

**Приложение Б**

|  |
| --- |
| struct Contact {  PublicKey organizationPublicKey  PublicKey userPublicKey  String userSignature  Integer timeframe  PrivateKey userPrivateKey = null  PublicKey confirmatorPublicKey = null }  contract SicknessRegistrar {  Map<String, Contact> contacts   registerContact(organizationPublicKey, userPublicKey, timeframe, userSignature) {  **if** actorRole != 'organization' **revert**  **if** actorPublicKey != organizationPublicKey **revert**  **if** timeframe != currentTimeFrame **revert**  contacts[userSignature] = new Contact(  organizationPublicKey,  userPublicKey,  timeframe,  userSignature  )  }  prune(){  **for**(key, value **in** contacts){  **if** value.timeframe < currentTimeframe - N {  contacts.remove(key)  }  }  }  } |

**Приложение В**

|  |
| --- |
| struct Contact {  PublicKey organizationPublicKey  PublicKey userPublicKey  String userSignature  Integer timeframe  PrivateKey userPrivateKey = null  PublicKey confirmatorPublicKey = null }  contract SicknessController {  Map<String, Contact> contacts   confirmContact(userSignature, timeframe, privateKey) {  **if** actorRole != 'medicalEmployee' **revert**  **if** not SicknessRegistrar.contacts.contains(userSignature) **revert**   contact = SicknessRegistrar.contacts[userSignature]  **if** contact.userPrivateKey != null **revert**  **if** contact.timeframe != timeframe **revert**  **if** encrypt(hash(contact.organizationPublicKey, contact.userPublicKey, timeframe), privateKey) != userSignature **revert**   contact.userPrivateKey = privateKey  contact.confirmatorPublicKey = actorPublicKey  organization = accessContract.organizations[contact.organizationPublicKey]  **if** organization == null **revert**   **if** organization.visits[timeframe] == null {  organization.visits[timeframe] = 0  }  organization.visits[timeframe] += 1  }   List<Integer> getSicknessRates(organizationPublicKey, timeframes){  organization = accessContract.organizations[organizationPublicKey]  **if** organization == null **revert**  rates = []  **for** timeframe **in** timeframes {  rates.add(organization.visits[timeframe])  }  **return** rates  } } |