**Модуль В. Аналитика.**

**Выполнил: Кирюхин А.С.**

**Отчет по синтаксису**

|  |  |  |
| --- | --- | --- |
| Строка | Ошибка | Исправление |
| 13 | wallet address; | address wallet; |
| 14 | whitelist bool; | bool whitelist; |
| 17 | enum Status [ Created,  Approved, Closed ] | enum Status { Created,  Approved, Closed } |
| 31 | require(msg.sender === owner, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); | require(msg.sender == owner, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); |
| 42 | whiteList[wallet] = WhiteData(address wallet, whiteList[wallet].whitelist); | whiteList[wallet] = WhiteData(wallet, whiteList[wallet].whitelist); |
| 41 | require(!whitelist[wallet].whitelist, "\_\_\_\_\_\_\_\_\_\_\_"); | require(!whiteList[wallet].whitelist, "\_\_\_\_\_\_\_\_\_\_\_"); |
| 35 | function createUser(string  \_login) public { | function createUser(string memory \_login) public { |
| 27-28 | require(products[product\_id].status == status, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");      } | require(products[product\_id].status == status, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          \_; |
| 46 | adr\_to.transfering(msg.value); | adr\_to.transfer(msg.value); |
| 61 | products.p(Product(products.lentgh, name, msg.sender, value, Status.Created, info)); | products.push(Product(products.length, name, msg.sender, value, Status.Created, info)); |
| 65 | products[product\_id].status = Status.Created(); | products[product\_id].status = Status.Created; |
| 83 | payable(owner).transfer(this.balance); | payable(owner).transfer(msg.value); |
| 82 | function withdrawFee() public onlyOwner { | function withdrawFee() public payable onlyOwner { |

**Отчет по функционалу**

Контракт написан на языке Solidity версии >0.5.0 и включает в себя 8 функций:

1. Создание пользователя. Принимает логин нового пользователя.
2. Обновление белого листа. Принимает адрес кошелька.
3. Отправить деньги. Принимает адрес, на который нужно отправить деньги, а также сумму.
4. Создание товара. Принимает название, цену и информацию о товаре.
5. Подтвердить товар. Принимает идентификатор товара.
6. Купить товар.Принимает идентификатор продукта.
7. Продать товар. Принимает идентификатор продукта.
8. Вывести деньги.

Помимо функций в контракте реализованы структуры данных:

1. Словарь пользователей
2. Словарь пользователей, входящих в белый список.
3. Перечисление статусов товара.
4. Структура для данных, входящих в белый список.
5. Структура товара.

Также контракт включает в себя конструктор, который делает хозяином контракта того, кто его развернул, а также модификаторы-проверки:

1. Проверяет, находится ли адрес в белом списке
2. Проверяет статус товара
3. Проверяет, вызывает функцию хозяин контракта или нет

**Отчет по логике**

|  |  |  |
| --- | --- | --- |
| Ошибка | Предложение по устранению ошибки | Пояснение |
| modifier checkOfWhiteLists(address adr) {          require(!whiteList[adr].whitelist, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          \_;      } | modifier checkOfWhiteLists(address adr) {          require(whiteList[adr].whitelist, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          \_;      } | Модификатор должен проверять, находится ли указанный адрес в белом списке или нет |
| function createUser(string memory \_login) public {          require(logins[\_login] == 0x0000000000000000000000000000000000000000, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          logins[\_login] =  address(0);      } | function createUser(address \_addr, string memory \_login) public {          require(logins[\_login] == 0x0000000000000000000000000000000000000000, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          logins[\_login] =  \_addr;      } | Функция должна создавать пользователя и по указанному логину присваивать ему указанный адрес. |
| function updateWhiteList(address wallet) public onlyOwner {          require(!whiteList[wallet].whitelist, "\_\_\_\_\_\_\_\_\_\_\_");          whiteList[wallet] = WhiteData(wallet, whiteList[wallet].whitelist);      } | function updateWhiteList(address wallet) public onlyOwner {          require(!whiteList[wallet].whitelist, "\_\_\_\_\_\_\_\_\_\_\_");          whiteList[wallet] = WhiteData(wallet, true);      } | Функция должна изменять белый список, добавляя туда адрес и статус, что этот адрес добавлен. |
| function approveProduct(uint product\_id) public checkStatusProduct(product\_id, Status.Created) onlyOwner {          products[product\_id].status = Status.Created;      } | function approveProduct(uint product\_id) public checkStatusProduct(product\_id, Status.Created) onlyOwner {          products[product\_id].status = Status.Approved;      } | Функция должна подтверждать товар для его продажи. |
| function createProduct(string memory name, uint value,  string memory info) public checkOfWhiteLists(msg.sender) {          products.push(Product(products.length, name, msg.sender, value, Status.Created, info));      } | Хозяин контракта автоматически не вносится в список, поэтому даже он не может создать товар.  Нужно внести его адрес в список при развертывании контракта в конструкторе (сл. ошибка) | Хозяин контракта должен сразу изначально входить в список, чтобы делать основные функции |
| constructor() {          owner = msg.sender;      } | constructor() {          owner = msg.sender;          whiteList[owner].whitelist = true;      } | Хозяин контракта должен сразу изначально входить в список, чтобы делать основные функции |
| function sellProduct(uint product\_id) public  checkStatusProduct(product\_id, Status.Closed) {          products[product\_id].status = Status.Approved;      } | function sellProduct(uint product\_id) public  checkStatusProduct(product\_id, Status.Closed) {          products[product\_id].status = Status.Closed;      } | Функция продажи товара должна устанавливать его статус на «Закрыт» |
| function buyProduct(uint product\_id) internal checkStatusProduct(product\_id, Status.Approved) { | function buyProduct(uint product\_id) public checkStatusProduct(product\_id, Status.Approved) { | Функция для покупки товара должна быть открыта для пользователей. |

**Отчет по проверкам**

|  |  |
| --- | --- |
| Текст в проверке | Предложение для текста |
| modifier checkOfWhiteLists(address adr) {          require(whiteList[adr].whitelist, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          \_;      } | The address is not included in white list! |
| modifier checkStatusProduct(uint product\_id, Status status) {          require(products[product\_id].status == status, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          \_;      } | The status of the product is different! |
| modifier onlyOwner() {          require(msg.sender == owner, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          \_;      } | This function must be called by owner! |
| function createUser(address \_addr, string memory \_login) public {          require(logins[\_login] == 0x0000000000000000000000000000000000000000, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");          logins[\_login] =  \_addr;      } | This login is already in white list! |
| function updateWhiteList(address wallet) public onlyOwner {          require(!whiteList[wallet].whitelist, "\_\_\_\_\_\_\_\_\_\_\_");          whiteList[wallet] = WhiteData(wallet, true);      } | This address is already in white list! |

**Отчет по информационной безопасности**

1. Отсутствие лицензии
2. Устаревшая версия Solidity
3. Версия Solidity указана не точно: >0.5.0

**Оптимизация смарт-контракта**

1. Исправить синтаксические ошибки.
2. Исправить логические ошибки.
3. В функции buyProduct убрать строку products[product\_id].status = Status.Closed; и заменить ее на вызов функции sellProduct, иначе можно убрать функцию sellProduct.
4. Функция sellProduct устанавливает статус товара как «Закрыт», но это можно сделать еще до покупки товара. В таком случае необходимо переименовать функцию на setProductClosed или на setProductStatus и доработать функционал.
5. Нарушена семантика форматирования согласно правилам языка Solidity. Параметры, которые принимают функции, должны начинаться с \_. Различия в отступах между строками и пробелами.
6. Название контракта не передает его смысл, функциональность и сущность.
7. В задании указано, что смарт-контракт был написан с использованием языка Solidity 0.8.0, в самом контракте указана версия >0.5.0
8. Объявленные структуры данных «разбросаны» по контракту и не находятся на своих местах согласно семантике языка Solidity.
9. Нарушение семантики языка Solidity: некоторые названия переменных и функций используют snake\_case в то время как в Solidity принят camelCase.
10. Отсутствие комментариев, поясняющих работу функций.