GrandPass.sol

1. Esiste una lista di address whitelisted, contenente una serie di indirizzi fissi di natura sconosciuta. La lista è stata lasciata invariata, ma sono stati aggiunti i seguenti metodi per modificarla dinamicamente:

```
function addWhitelistUser(address user) external onlyOwner {
    require(user != address(0), "Invalid address");
    whitelisted[user] = true;
}
function removeWhitelistUser(address user) external onlyOwner {
    require(whitelisted[user], "User not in whitelist");
    whitelisted[user] = false;
}
```

2. Viene utilizzato un riferimento AggregatorV3Interface(0x71041dddad3595F9CEd3DcCFBe3D1F4b0a16Bb70). L'indirizzo è un riferimento al contratto Chainlink per il prezzo ETH/USD su Binance Smart Chain (BSC).
    - Non volendo modificare la signature del costruttore, è stato aggiunto un metodo per modificare l'indirizzo:

```
// Funzione per aggiornare l'indirizzo del data feed
function setDataFeedAddress(address newAddress) external onlyOwner {
    require(newAddress != address(0), "Invalid address");
    dataFeedAddress = newAddress;
    dataFeed = AggregatorV3Interface(newAddress); // Aggiorna anche l'istanza dell'interfaccia
}
```

3. L'uso del prezzo del data feed è utilizzato solo per la generazione di un numero casuale e potrebbe essere rimosso per rendere il codice più facile da trasportare su diverse chain.
4. Questo contratto permette di mintare un massimo di **119 NFT** riferiti a **20 RWA** differenti, ma non è chiaro lo scopo.

GToken.sol

Questo contratto implementa un token ERC20 con controllo centralizzato sulla creazione e distruzione dei token, ideale per scenari come un **token collateralizzato**, in cui un amministratore (o un insieme di entità autorizzate) deve avere il controllo su **minting** e **burning**.

1. Il significato di underlying non è immediatamente chiaro dal contesto; potrebbe essere un **placeholder** per un indirizzo di un asset sottostante in un utilizzo futuro, ma in questo stato attuale

non viene utilizzato attivamente nel contratto. underlying rappresenta probabilmente l'indirizzo del token o dell'asset che funge da **collaterale** per GToken.

Processo di Collateralizzazione

- **Deposito Collaterale**: Un utente deposita una certa quantità di token underlying nel contratto GToken.
- **Minting di GToken**: In cambio del collaterale depositato, vengono mintati nuovi GToken all'indirizzo dell'utente.
- **Burning e Riscatto**: Quando l'utente desidera ritirare il collaterale, deve restituire una quantità equivalente di GToken. I GToken vengono bruciati e il collaterale viene restituito all'utente.

MasterGrand.sol

MasterGrand sembra ispirato al contratto MasterChef di SushiSwap ed è utilizzato per gestire l'allocazione di ricompense in **GrandToken**.

Utilizzo del Contratto

- **Farming e Staking**: Questo contratto permette di creare un meccanismo di **farming** e **staking**, in cui gli utenti possono depositare token di liquidità (LP) e ricevere **GrandToken** come ricompensa.
- **Incentivazione tramite GrandToken**: GrandToken viene utilizzato come incentivo per gli utenti a fornire liquidità o a contribuire al protocollo, premiando i partecipanti con nuovi token.

Opzioni di Deposito

- In MasterGrand possono essere depositati sia **token LP** che **GToken**, a seconda della configurazione del contratto e degli obiettivi del protocollo. Vediamo due scenari:

    1. **Uso di GToken come LP Token**:
        - Gli utenti possono depositare collaterale e ricevere GToken, utilizzabili come **LP token** nel contratto MasterGrand per guadagnare ricompense in GrandToken.
        - Questo approccio consente agli utenti di ottenere ulteriori ricompense GrandToken per il collaterale depositato.
    2. **Deposito di Token LP di Uniswap/Aerodrome**:
        - Gli utenti possono fornire liquidità su piattaforme come **Uniswap** o **Aerodrome**, ricevere token LP, e metterli in staking in MasterGrand per guadagnare ricompense aggiuntive.

Criticità

- La funzione massUpdatePools potrebbe diventare molto costosa in termini di **gas** se ci sono molti pool, dato che deve aggiornare ciascun pool individualmente.
- Il contratto menziona un **collector**, ma questo non sembra essere utilizzato attivamente. Potrebbe essere pianificato come un destinatario di una parte delle ricompense o come una fee di gestione.

Modifiche

- Sistemato il costruttore e importato correttamente @openzeppelin/contracts/token/ERC20/ERC20.sol.

PotionVault.sol (Sistema simile ai Vault di MakerDAO)

Modifiche

- Sistemato il costruttore.
- Modificata la funzione mintFeeRatio, il cui nome sovrascriveva la variabile omonima:

```
function setMintFeeRatio(uint _mintFeeRatio) external onlyOwner {
   require(_mintFeeRatio > 0, "zero is invalid");
   mintFeeRatio = _mintFeeRatio;
}
```

- **TODO**: Risolvere il problema della **dimensione massima** del codice (27144 Byte, limite massimo 24576 Byte). Soluzioni possibili:
    - Abilitare l'**Optimizer di Solidity**:
        - **Truffle**: Configurare truffle-config.js: compilers (runs valore basso, es. 100 o 200): {solc: {version: "0.8.28", settings: {optimizer: {enabled: true, runs: 200 }}}}
        - **Remix**: Advanced Configurations > Enable optimizer e impostare runs a un valore basso (es. 100 o 200).

Funzioni Principali del Contratto

- **openSafe**: Permette agli utenti di aprire una nuova **Safe** depositando collaterale e mintando GToken.
- **closeSafe**: Permette all'utente di **chiudere una Safe**, bruciando i GToken e ritirando il collaterale.
- **depositCollateral** e **withdrawCollateral**: Permettono rispettivamente di aggiungere o ritirare collaterale da una Safe.
- **mint** e **burn**: Permettono di **mintare** nuovi GToken o **bruciare** quelli esistenti per ridurre il debito.
- **liquidate**: Consente la **liquidazione** delle Safe non conformi ai requisiti di collateralizzazione.

Incentivi

- **Commissioni per i Liquidatori**: I liquidatori ricevono una parte del collaterale come compenso per aver ripristinato la sicurezza del sistema.

XGrand.sol

Il contratto XGrand ha lo scopo di permettere agli utenti di **depositare i token Grand** per ottenere **xGrand** in cambio, rappresentando una partecipazione allo staking di Grand.

Da chiarire:

- **underlying**: Attualmente non viene utilizzato, ma potrebbe rappresentare un asset sottostante in un contesto futuro.
- **grandAddress**: Riferimento al token Grand passato nel costruttore. Potrebbe riferirsi al contratto **Grand.sol** nella cartella testnet.