🔥 DUMP.TIRES - DTGC Staking Platform

> PulseChain DeFi Staking for Diamond Hands 💎

Show Image

Show Image

🌐 Live Site

dump.tires - Connected to pump.tires pre-launched coins

📊 Tokenomics

Metric	Value
Total Supply	1,000,000,000 DTGC
Dev Wallet	830,000,000 (83%)
DAO Rewards	200,000,000 (20%)
Circulating	100,000,000 (10%)

Token Contracts (PulseChain)

Token	Address
DTGC	0xD0676B28a457371D58d47E5247b439114e40Eb0F
URMOM	0xe43b3cEE3554e120213b8B69Caf690B6C04A7ec0

Token	Address
Burn Address	0x0000000000000000000000000000000000000369

🎯 Features

Staking System

Tiered APR with Diamond Hands Bonus:

Tier	Lock Period	Base APR	Diamond Bonus	Total Potential
🥉 Bronze	14 days	5%	+1%	6%
🥈 Silver	30 days	12%	+3%	15%
💎 Diamond	90 days	25%	+8%	33%

Fee Structure:

- **Entry Fee:** 5% (50% to staker pool, 50% to DAO)
- **Exit Fee:** 5% (normal withdrawal after lock)
- **Emergency Penalty:** 20% (early withdrawal)

Reward Mechanisms

1. **Base APR Rewards** - Linear rewards based on stake duration
2. **Fee Share Rewards** - Share of entry fees from new stakers
3. **Diamond Hands Bonus** - Extra rewards for completing lock period

DAO Treasury

- 200M DTGC allocated for community rewards
- Proposal system for fund distribution
- Multi-sig council for governance
- Daily spending limits for security

🏗️ Architecture

```
dump-tires/
├── contracts/
│   ├── DTGCStaking.sol      # Main staking contract
│   └── DTGCDAOTreasury.sol  # DAO treasury & governance
├── frontend/
│   ├── src/
│   │   ├── App.jsx          # Main React app
│   │   ├── config/          # Chain & contract config
│   │   ├── components/      # UI components
│   │   ├── hooks/           # Custom React hooks
│   │   └── utils/           # Helper functions
│   └── package.json
└── scripts/
    └── deploy.js            # Deployment scripts
```

🚀 Deployment Guide

Prerequisites

- Node.js 18+
- Hardhat or Foundry
- PulseChain RPC access
- PLS for gas fees

1. Deploy Contracts

```bash
```

```
# Install dependencies
cd contracts
npm install

# Configure .env
cp .env.example .env
# Add: PRIVATE_KEY, PULSECHAIN_RPC

# Deploy DAO Treasury first
npx hardhat run scripts/deploy-treasury.js --network pulsechain

# Deploy Staking with treasury address
npx hardhat run scripts/deploy-staking.js --network pulsechain

# Verify contracts
npx hardhat verify --network pulsechain <STAKING_ADDRESS> <DTGC_ADDRESS> <TREASURY_ADDRESS> <REV
```

2. Fund the Contracts

solidity

```
// From dev wallet, approve and fund treasury
dtgcToken.approve(treasuryAddress, 200_000_000 * 1e18);
treasury.fundTreasury(200_000_000 * 1e18);

// Approve staking contract in treasury
treasury.approveStakingContract(stakingAddress);

// Fund staking rewards
treasury.fundStakingContract(stakingAddress, 50_000_000 * 1e18);
```

3. Deploy Frontend

bash

```
cd frontend

# Install dependencies
npm install

# Update contract addresses in src/config/constants.js

# Build for production
npm run build

# Deploy to your hosting (Vercel, Netlify, etc.)
```

🔧 Contract Interaction

Staking

```javascript
import { ethers } from 'ethers';

const stakingContract = new ethers.Contract(STAKING_ADDRESS, STAKING_ABI, signer);

// Approve DTGC spending
await dtgcToken.approve(STAKING_ADDRESS, amount);

// Stake with tier selection (0=14d, 1=30d, 2=90d)
await stakingContract.stake(amount, 2); // Diamond tier

// Check position
const position = await stakingContract.getPosition(userAddress);

// Calculate pending rewards
const [base, feeShare, diamond] = await stakingContract.calculateAllRewards(userAddress);

// Claim rewards
await stakingContract.claimRewards();

// Withdraw after lock
await stakingContract.withdraw();
```

DAO Treasury

```javascript
const treasuryContract = new ethers.Contract(TREASURY_ADDRESS, TREASURY_ABI, signer);

// Check budgets
const [staking, lp, community, ops] = await treasuryContract.getBudgets();

// Create proposal
await treasuryContract.createProposal(
  "Fund staking rewards Q1 2025",
  stakingAddress,
  ethers.parseEther("10000000"),
  0 // ProposalType.FUND_STAKING
);

// Approve proposal (council members)
await treasuryContract.approveProposal(1);

// Execute proposal
await treasuryContract.executeProposal(1);
```

🔒 Security Considerations

1. **Timelock on Admin Functions** - Consider adding timelock for critical operations
2. **Multi-sig Guardian** - Replace single guardian with Gnosis Safe
3. **Audit Recommendations** - Get professional audit before mainnet
4. **Emergency Pause** - Add pausable functionality for emergencies
5. **Upgrade Path** - Consider proxy pattern for upgradeability

---

🔗 Integration with pump.tires

The dump.tires platform is directly linked to pump.tires pre-launched coins:

- **URMOM/DTGC PLP Pair** (Coming Soon)

- Burned LP tokens viewable at: `0x000000000000000000000000000000000000369`
- Cross-platform rewards for ecosystem participants

📈 Future Roadmap

- [x] Core staking contracts
- [x] DAO Treasury with governance
- [x] Frontend dApp
- [ ] URMOM/DTGC liquidity pair
- [ ] LP staking rewards
- [ ] Governance token voting
- [ ] NFT badges for long-term stakers
- [ ] Mobile app

📜 License

MIT License - see LICENSE for details

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open Pull Request

⚠️ Disclaimer

This is experimental DeFi software. Use at your own risk. Not financial advice. Always DYOR (Do Your Own Research).

Built with 💜 on PulseChain