🧠 DAOkit on Gno.land

Modular Governance for Gno Smart Contracts

Building decentralized organizations with composable components

🧠 Use Case: Funding a Project

Outcome:

- 📅 If approved before the deadline: Project is funded
- ❌ Otherwise: Proposal is rejected

🧩 DAOkit Components

📦 DAOkit

- Core package: **Proposals**, **Resources**

🧱 basedao

- Manages **Members** and **Roles**

⚙️ daocond

- Handles **Conditions** for proposal execution

🛠️ Implementing DAOkit

🧍 Adding Roles and Members

```
initialRoles := []basedao.RoleInfo{
    {Name: "admin", Description: "Admin is the superuser"},
    {Name: "public-relationships", Description: "Handles public communication"},
    {Name: "finance-officer", Description: "Manages funds"},
}

initialMembers := []basedao.Member{
    {Address: "g126...zlg", Roles: []string{"admin", "public-relationships"}},
    {Address: "g1ld6...3jv", Roles: []string{"public-relationships"}},
    {Address: "g1r69...0tth", Roles: []string{"finance-officer"}},
    {Address: "g16jv...6e0r", Roles: []string{}},
}
```

⚙️ DAO Initialization

```
var (
    DAO        daokit.DAO
    daoPrivate *basedao.DAOPrivate
)

func init() {
    memberStore := basedao.NewMembersStore(initialRoles, initialMembers)

    condition := daocond.And(
        daocond.MembersThreshold(0.6, memberStore.IsMember, memberStore.MembersCount),
        daocond.RoleCount(1, "finance-officer", memberStore.HasRole),
    )

    DAO, daoPrivate = basedao.New(&basedao.Config{
        Name:             "Demo DAO",
        Description:      "A demo DAO built with DAOkit",
        Members:          memberStore,
        InitialCondition: condition,
    })
}
```

🗳️ Voting and Execution

```
func Vote(proposalID uint64, vote daocond.Vote) {
    DAO.Vote(proposalID, vote)
}

func Execute(proposalID uint64) {
    DAO.Execute(proposalID)
}

func Propose(req daokit.ProposalRequest) {
    DAO.Propose(req)
}

func Render(path string) string {
    return daoPrivate.Render(path)
}
```

🔧 Create new Condition

```
type Condition interface {
    Eval(votes map[string]Vote) bool
    Signal(votes map[string]Vote) float64
    Render() string
    RenderWithVotes(votes map[string]Vote) string
}
```

📦 Create new Resource

```
type Action interface {
    String() string
    Type() string
}

type ActionHandler interface {
    Execute(action Action)
    Type() string
}
```