



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

- 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

```
1  var (  
2      DAO          daokit.DAO  
3      daoPrivate *basedao.DAOPrivate  
4  )  
5  
6  func init() {  
7      memberStore := basedao.NewMembersStore(initialRoles, initialMembers)  
8  
9      condition := daocond.And(  
10         daocond.MembersThreshold(0.6, memberStore.IsMember, memberStore.MembersCount),  
11         daocond.RoleCount(1, "finance-officer", memberStore.HasRole),  
12     )  
13  
14     DAO, daoPrivate = basedao.New(&basedao.Config{  
15         Name:          "Demo DAO",  
16         Description:    "A demo DAO built with DAOkit",  
17         Members:        memberStore,  
18         InitialCondition: condition,  
19     })  
20 }
```

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.Produce(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  
}
```