One Backend to ~~Rule~~ Serve Them All

Multi-Tenant Customization

Garth Henson, Director of Engineering

One Ring to rule them all,
One Ring to find them,
One Ring to bring them all
and in the darkness bind them

- JRR Tolkien

```
Parent Company
├── Sports Division
|   ├── Game Alerts
|   ├── Team Preferences
|   └── Statistics Updates
├── Entertainment Division
|   ├── Content Preferences
|   ├── Viewing History
|   └── Recommendations
└── Parks Division
    ├── Trip Planning
    ├── Dining Preferences
    └── Experience Settings
```

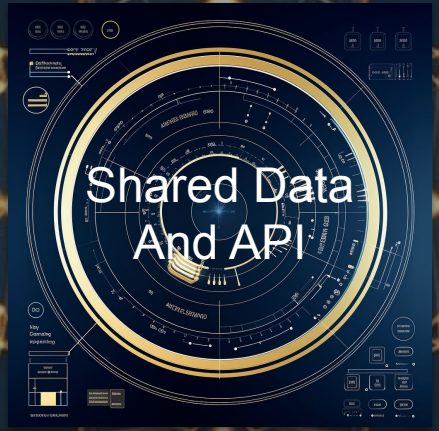
🔄 Multiple Systems

🎨 Brand Identity

👤 User Experience

📊 Data Consistency

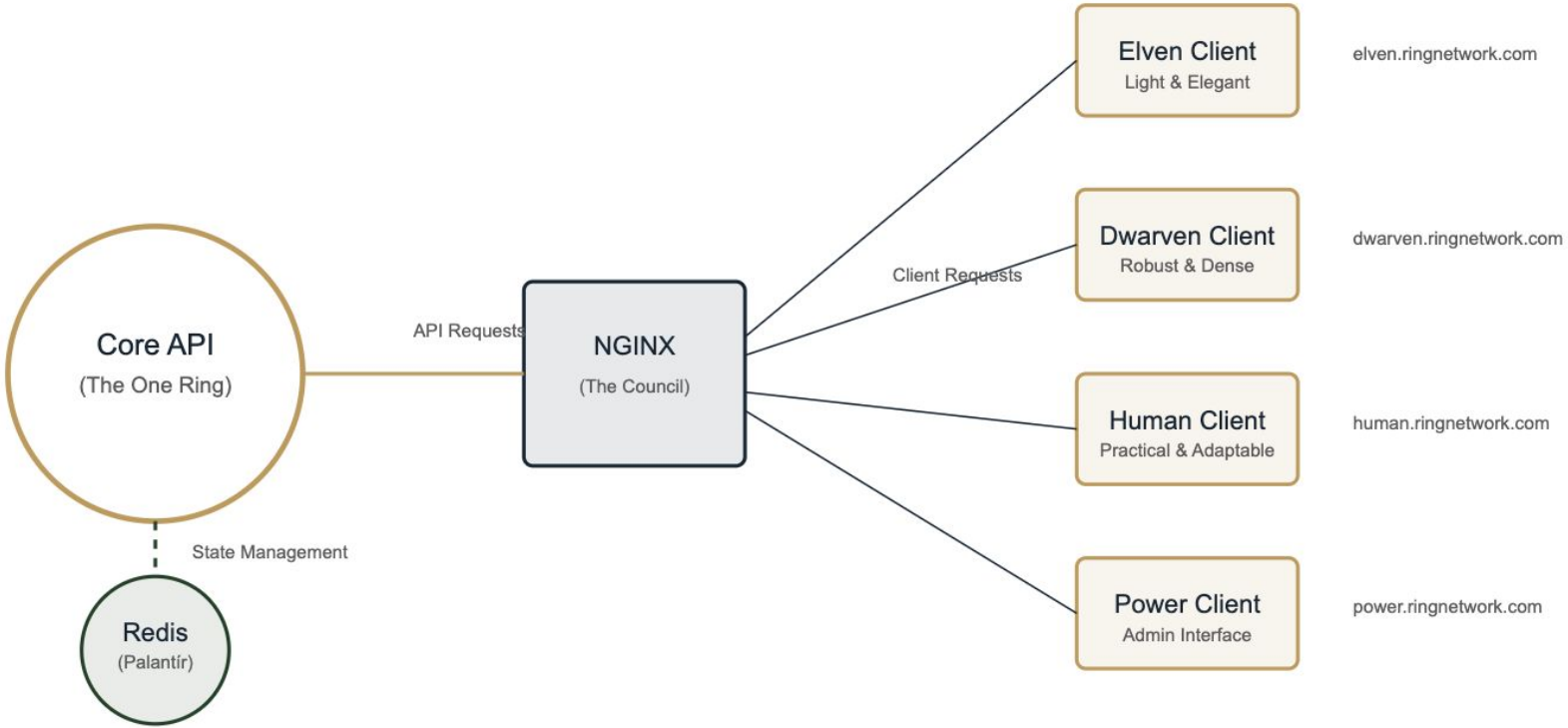

Elven Client


Shared Data
And API


Power Client


Dwarven Client


Human Client


Core API
(The One Ring)
API Requests
NGINX
(The Council)
Client Requests
Elven Client
Light & Elegant
elven.ringnetwork.com
Dwarven Client
Robust & Dense
dwarven.ringnetwork.com
Human Client
Practical & Adaptable
human.ringnetwork.com
Power Client
Admin Interface
power.ringnetwork.com
State Management
Redis
(Palantír)

NGINX (The Council)

```
# Example configuration snippet
server {
    listen 80;
    server_name dwarven.ringnetwork.com;
    location / {
        proxy_set_header X-Client-ID "dwarven";
        proxy_pass http://dwarven-client;
    }
}
```

Redis (The Palantír)

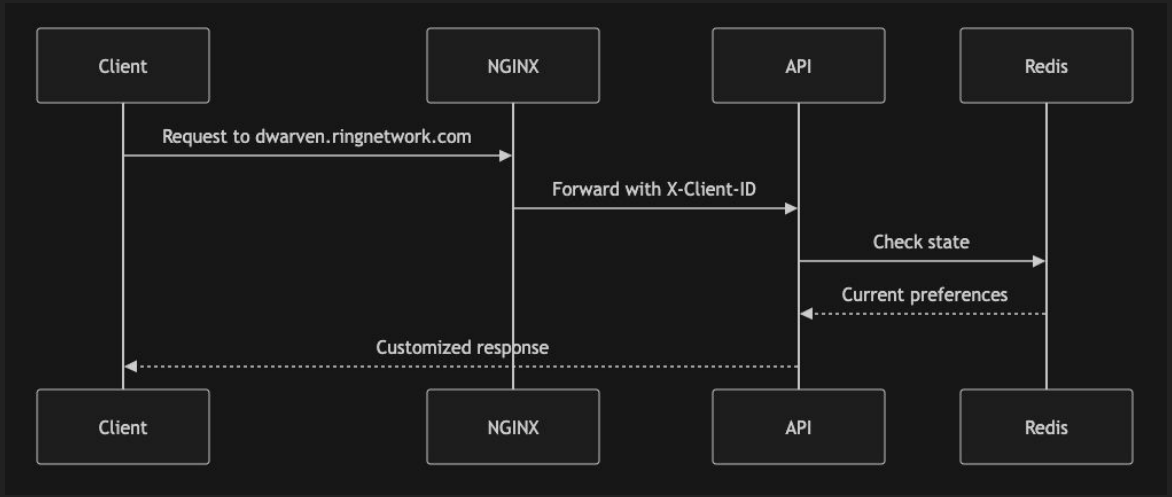
```
class StateManager {
    async updatePreference(key: string, value: any): Promise<void>;
    async notifyRealms(update: PreferenceUpdate): Promise<void>;
}
```

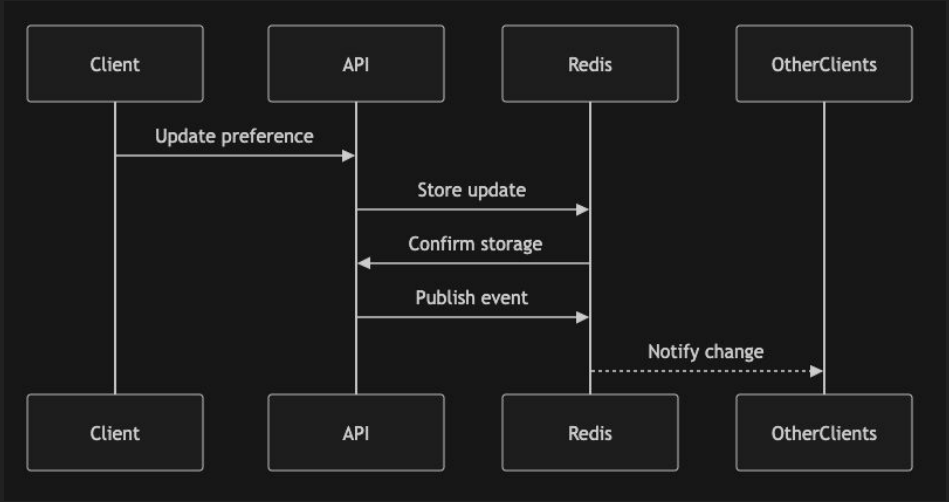
Core API (The One Ring)

```
interface PreferenceManager {
    getGlobalPreferences(): Promise<Preference[]>;
    getRealmPreferences(realmId: string): Promise<Preference[]>;
}
```

Client Realms

```
interface RealmConfig {
    themeOverrides: ThemeOptions;
    customPreferences: PreferenceDefinition[];
}
```


Client
NGINX
API
Redis
Request to dwarven.ringnetwork.com
Forward with X-Client-ID
Check state
Current preferences
Customized response
Client
NGINX
API
Redis


Client
API
Redis
OtherClients
Update preference
Store update
Confirm storage
Publish event
Notify change
Client
API
Redis
OtherClients

Realms Demo

Middle-Earth Preference Management

Implementation Checklist: Infrastructure Foundation

Essential Setup

- Domain strategy for each tenant
- NGINX or similar reverse proxy
- Redis or alternative state store
- Container orchestration (Docker / K8s)

Key Considerations

- Scalability requirements
- SSL certificate management
- Development environment parity
- Monitoring requirements

Implementation Checklist: Preference Architecture

Core Design

- Global preference schema
  - Required fields
  - Default values
  - Validation rules
- Tenant-specific extensions
  - Override mechanisms
  - Custom fields
  - Validation rules

Data Management

- State synchronization strategy
- Cache invalidation patterns
- Backup and recovery plans
- Migration paths

Implementation Checklist: Multi-Tenant Implementation

Isolation Patterns

- Request routing
- Header injection
- State partitioning
- Error boundaries

Security Measures

- Authentication per tenant
- Authorization scoping
- Data segregation
- Audit logging

Implementation Checklist: Client Integration

Interface Design

- Global component library
- Theme management
- Override patterns
- Responsive strategies

State Management

- Real-time updates
- Offline capabilities
- Conflict resolution
- Error handling

Implementation Checklist: Administrative Capabilities

Management Interface

- Cross-tenant visibility
- Audit capabilities
- Configuration management
- User management

Monitoring

- Health checks
- Performance metrics
- Usage analytics
- Alert systems

Additional Considerations

Common Pitfalls to Avoid

- Over-coupling tenant implementations
- Insufficient state management
- Inadequate error boundaries
- Poor performance monitoring
- Weak testing strategy

Scaling Guidelines

- Horizontal scaling preparation
- Database partitioning strategy
- Cache distribution approach
- Monitoring and alerting setup

Performance Considerations

- Cache strategy implementation
- State synchronization patterns
- Resource isolation
- Load balancing configuration

Implementation Timeline

Phase 1: Foundation

- Infrastructure setup
- Core preference architecture
- Basic tenant isolation

Phase 2: Features

- Client implementations
- State management
- Administrative interface

Phase 3: Production

- Security hardening
- Performance optimization
- Monitoring implementation

Success Metrics

- Response time targets
- State sync latency
- Resource utilization
- Error rates

Demo Code

https://github.com/guahanweb/demo-power-network

Thank You!

Please leave feedback