agenda

1. Vorstellung des Teams

- Kurze Teamvorstellung
- Kernziele des Projekts
- Projektüberblick in einem Satz

2. Terraform & Infrastructure as Code

Kurze Terraform Einführung
Unsere AWS-Infrastruktur
Wichtigste Komponenten
Automatisierung
Live-Einblick in Code-Beispiel

3. Frontend-Entwicklung

Tech-Stack Überblick
React + Vite
Mehrsprachigkeit
Wichtigste Features zeigen
Komponenten-Struktur
Authentifizierung

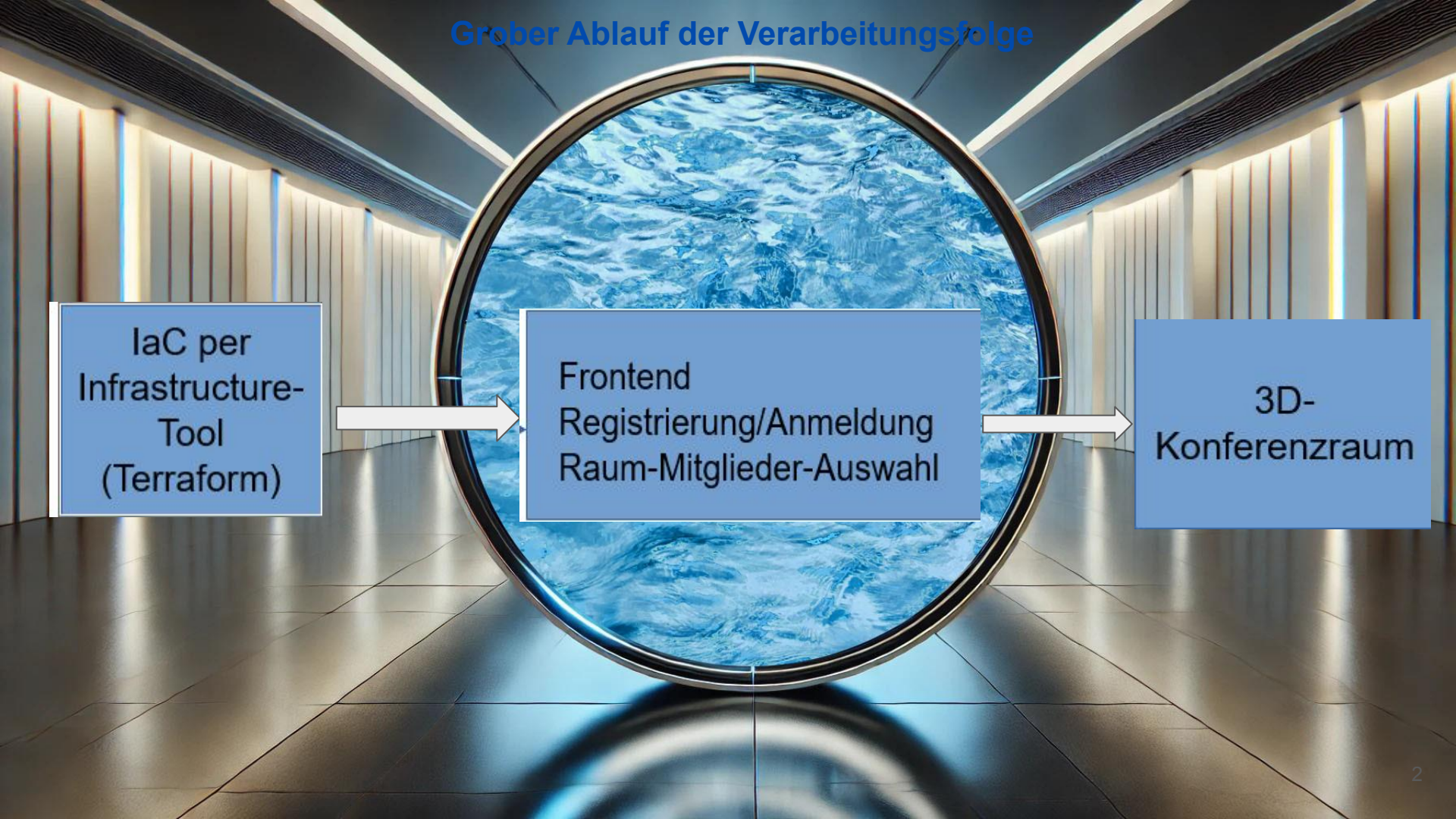
4. Live Demo

Schnelldurchlauf der Hauptfunktionen
Login/Registrierung
Sprachauswahl
Raumansicht

5. Abschlusswort

Kern Erkenntnisse
Nächste Schritte
Danke

Grober Ablauf der Verarbeitungsfolge



🌟 Teilprojekt Terraform Infrastruktur📖 Projektübersicht

Dieses Teil-Projekt wurde entwickelt, um eine vollständig modulare, dynamische und skalierbare Cloud-Infrastruktur mithilfe von Terraform zu implementieren. Ziel ist es, die Verwaltung der Infrastruktur effizient zu gestalten und gleichzeitig Best Practices für Skalierbarkeit, Sicherheit und Wartbarkeit einzuhalten.

Über das aktuelle Projekt hinaus wird dieser Teil eigenständig weiterentwickelt.

Alle Konfigurationen werden zentral über eine Datei (`terraform.tfvars`) gesteuert, um Anpassungen einfach und konsistent vorzunehmen.

🚀 Funktionsweise

1. **Modulare Struktur**

•Verzeichnisstruktur:

1.Module befinden sich im Verzeichnis `/modules/`.

2.Die Hauptdateien wie `main.tf`, `variables.tf` und `outputs.tf` rufen Module auf und definieren globale Konfigurationen.

Vorteile:

1.Wiederverwendbarkeit der Module.

2.Klare Trennung der Verantwortlichkeiten.

2. Zentrale Konfiguration

•Alle Einstellungen werden in der Datei `terraform.tfvars` vorgenommen.

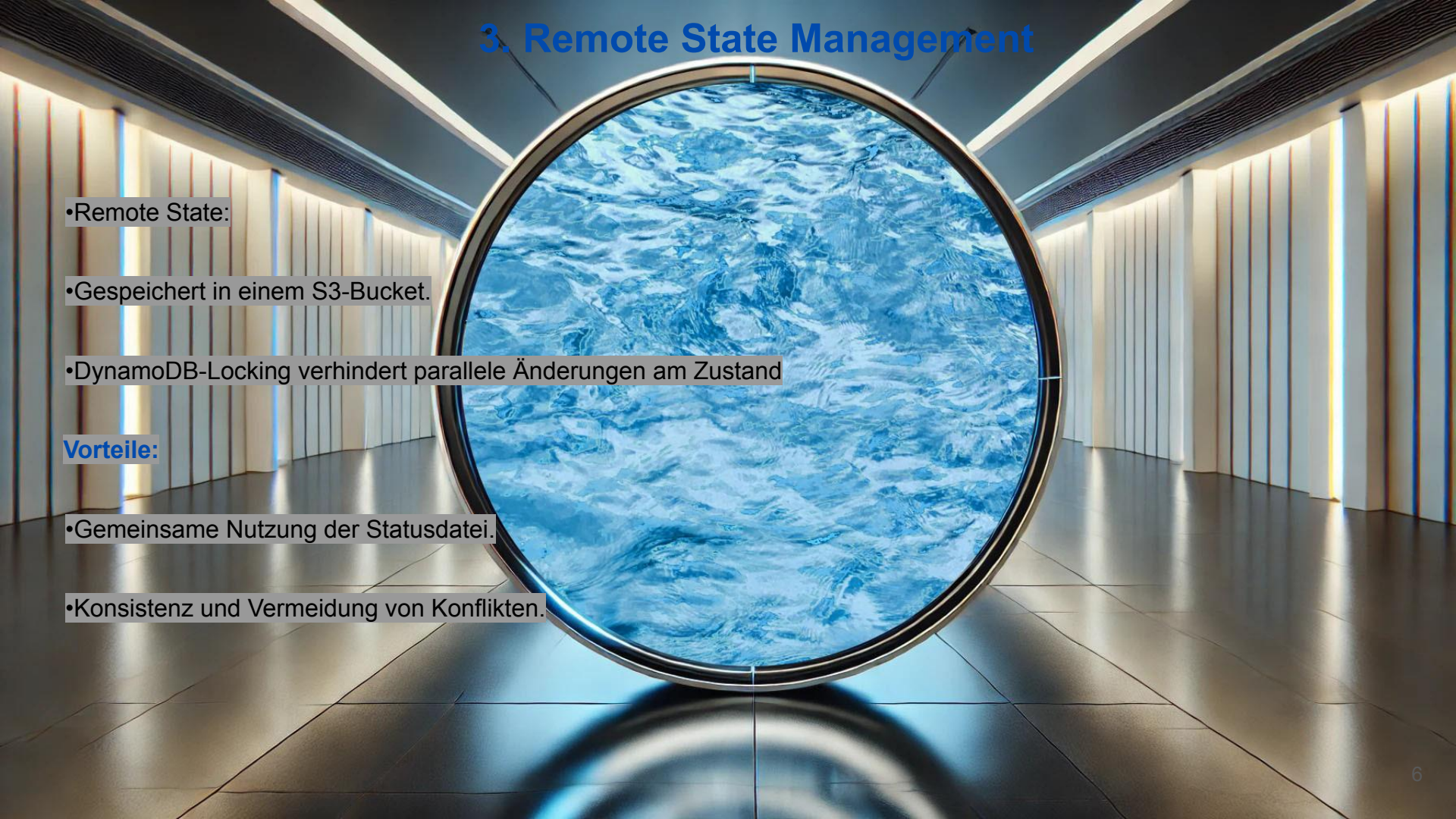
Beispiele:

•Ressourcen-Namen (z. B. S3-Buckets, SNS-Themen).

•Schwellenwerte für CloudWatch-Alarme.

•Zugangsdaten und Einstellungen für Cognito.

3. Remote State Management



•Remote State:

•Gespeichert in einem S3-Bucket.

•DynamoDB-Locking verhindert parallele Änderungen am Zustand

Vorteile:

•Gemeinsame Nutzung der Statusdatei.

•Konsistenz und Vermeidung von Konflikten.

🛠️ Schritte zur Verwendung

1. Voraussetzungen

•Installierte Tools:

1.Terraform (Version >= 1.5.0).

2.AWS CLI (mit konfigurierten Zugangsdaten).

•AWS Berechtigungen:

1.Zugriff auf S3, DynamoDB, CloudWatch, SNS und Cognito und ggf. weitere Services.

✨ Best Practices

Modularisierung

•Jedes Infrastruktur-Element wird als Modul implementiert.

•Fördert Wiederverwendbarkeit und Übersichtlichkeit.

Dynamische Konfiguration

•Zentrale Steuerung über die Datei `terraform.tfvars`.

Sicherheitsmaßnahmen

•Private Schlüssel haben restriktive Berechtigungen (`chmod 600`).

•Keine öffentlichen S3-Buckets.

Remote State

•S3 für den Zustand.

•DynamoDB für Locking zur Vermeidung paralleler Änderungen.

🛠️ Fehlerbehebung

Remote State Fehler

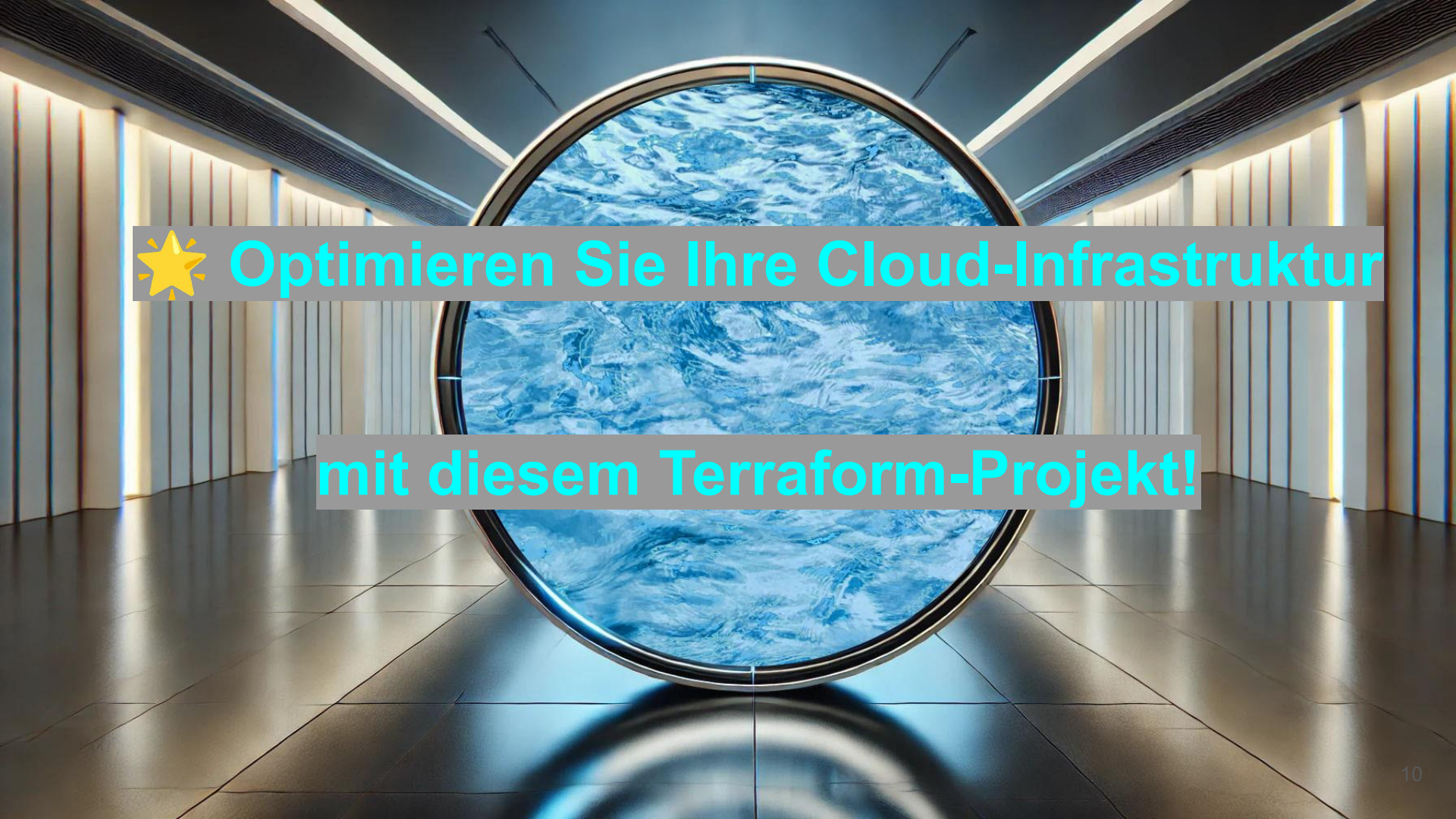
•Überprüfen, ob der S3-Bucket und die DynamoDB-Tabelle korrekt konfiguriert sind.

AWS Berechtigungen

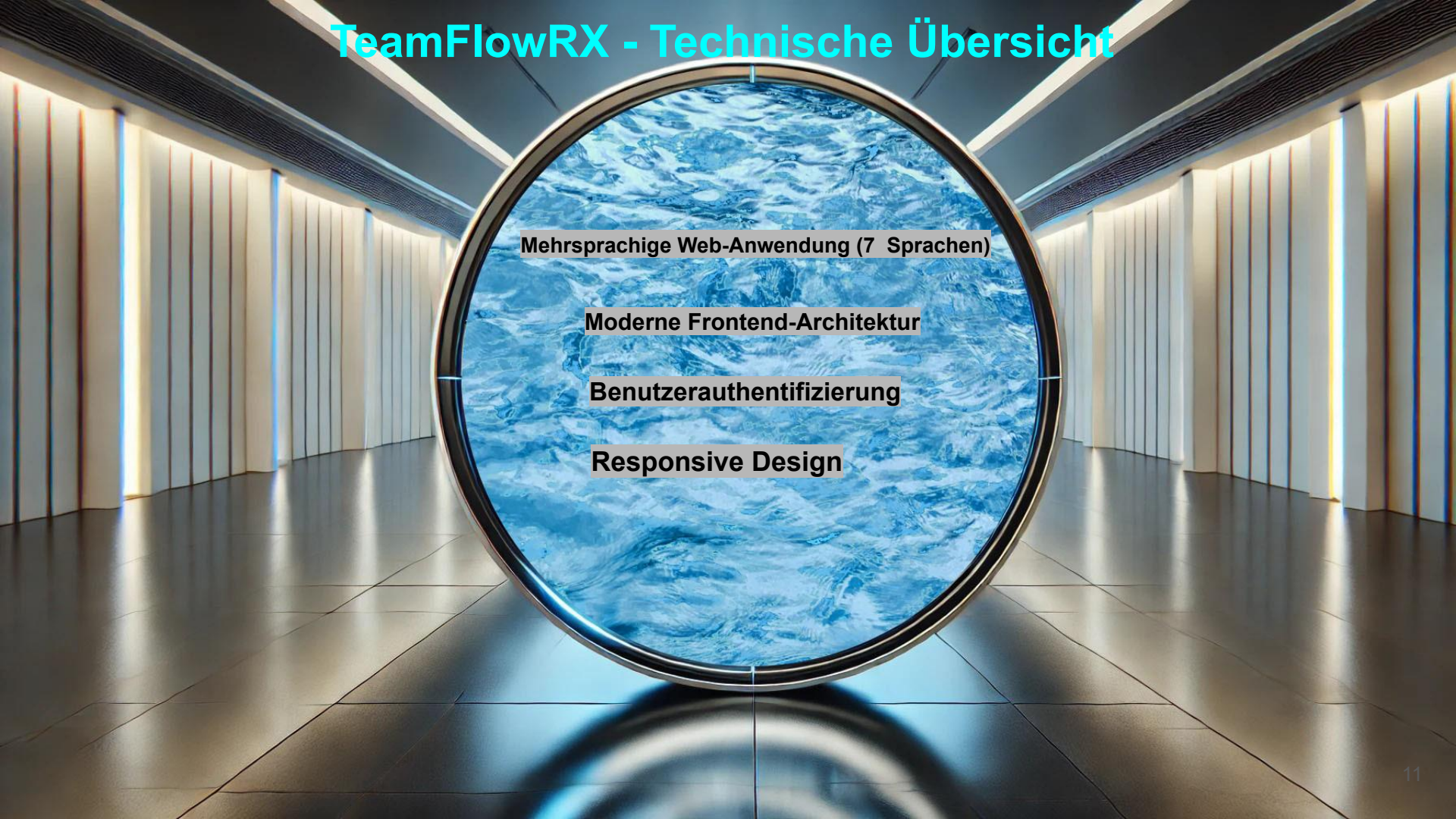
•Sicherstellen, dass alle erforderlichen Berechtigungen vorhanden sind.

Modulfehler

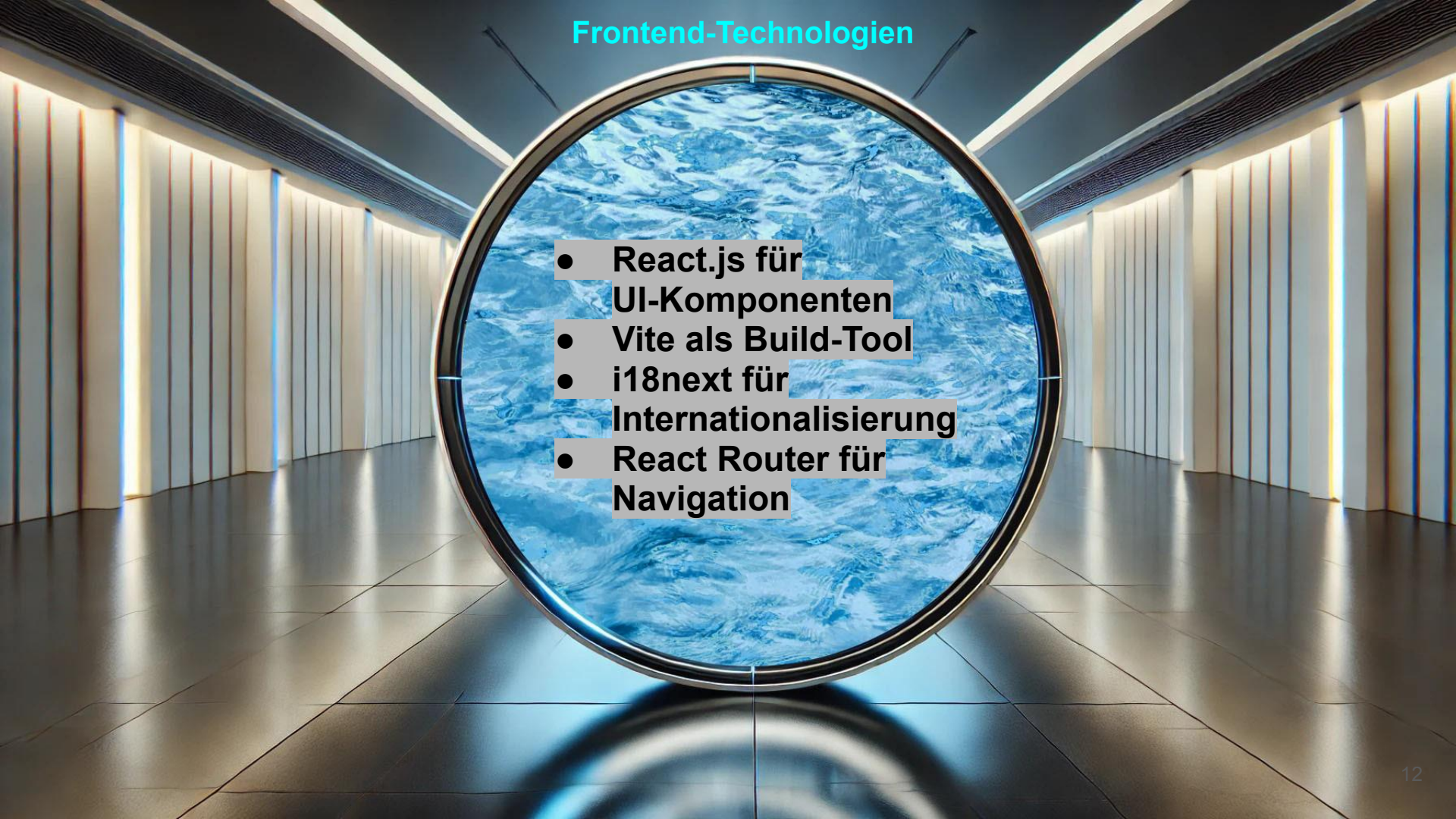
•Vergewissern, dass alle Module im Verzeichnis `/modules/` korrekt liegen.


🌟 Optimieren Sie Ihre Cloud-Infrastruktur
mit diesem Terraform-Projekt!

TeamFlowRX - Technische Übersicht

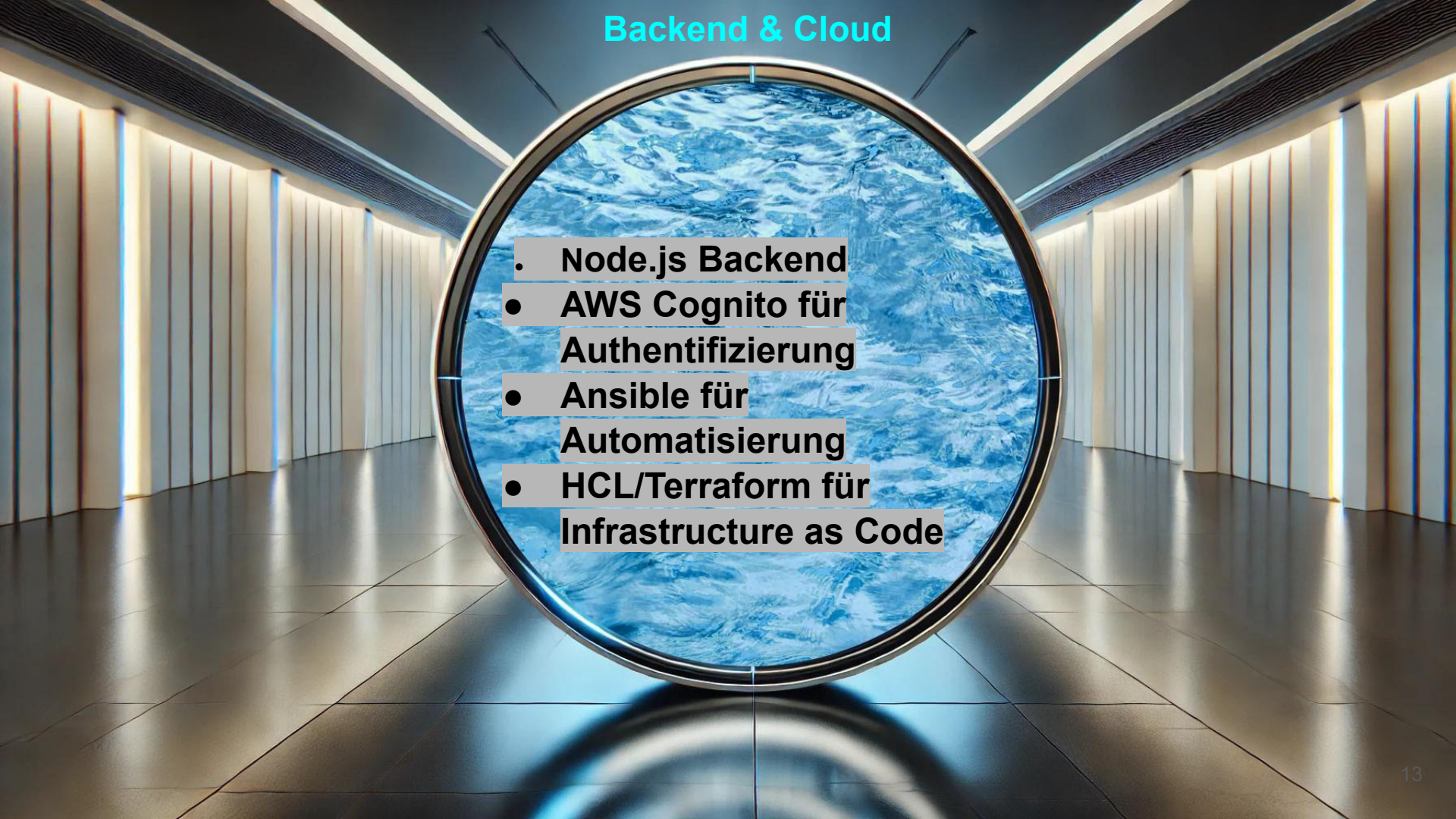


Frontend-Technologien



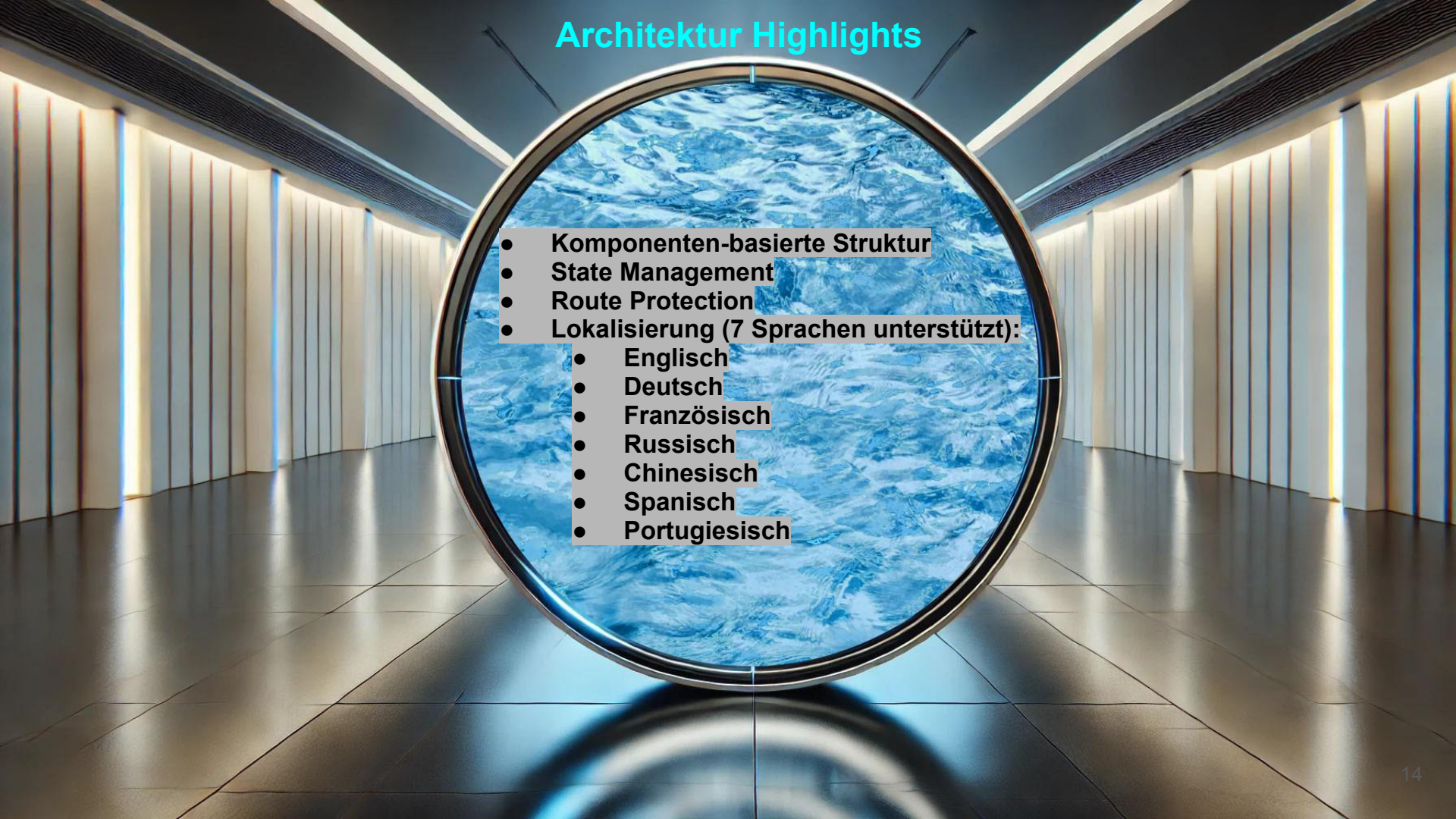
- React.js für UI-Komponenten
- Vite als Build-Tool
- i18next für Internationalisierung
- React Router für Navigation

Backend & Cloud



- Node.js Backend
- AWS Cognito für Authentifizierung
- Ansible für Automatisierung
- HCL/Terraform für Infrastructure as Code

Architektur Highlights

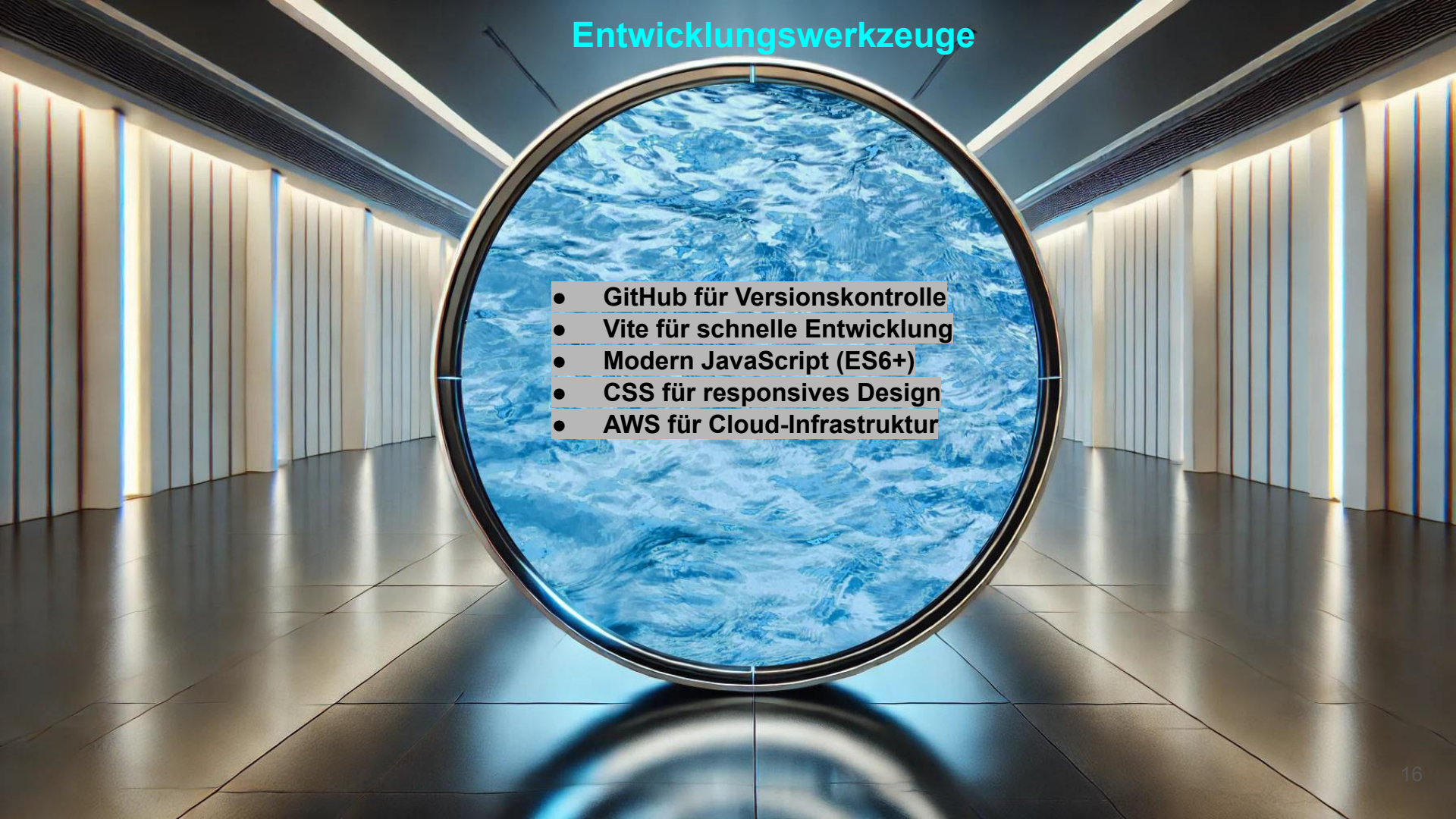


- Komponenten-basierte Struktur
- State Management
- Route Protection
- Lokalisierung (7 Sprachen unterstützt):
  - Englisch
  - Deutsch
  - Französisch
  - Russisch
  - Chinesisch
  - Spanisch
  - Portugiesisch

Besondere Features

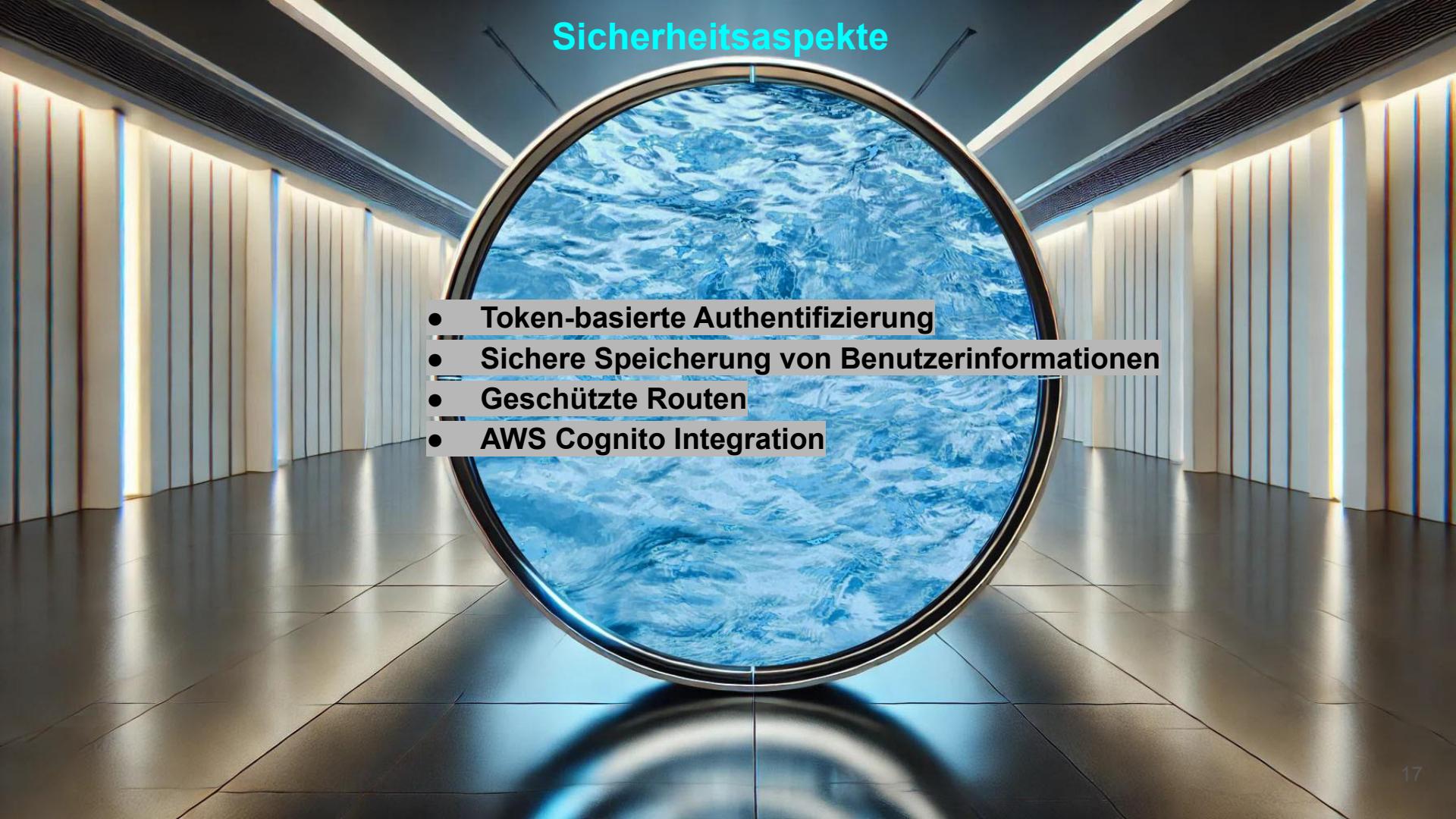
- Dynamisches Sprachmanagement
- Persistente Spracheinstellungen
- Video-Hintergrund Integration
- Responsives Layout
- Geschützte Routen
- Benutzerauthentifizierung
- Session Management

Entwicklungswerkzeuge



- GitHub für Versionskontrolle
- Vite für schnelle Entwicklung
- Modern JavaScript (ES6+)
- CSS für responsives Design
- AWS für Cloud-Infrastruktur

Sicherheitsaspekte



- Token-basierte Authentifizierung
- Sichere Speicherung von Benutzerinformationen
- Geschützte Routen
- AWS Cognito Integration