Documentazione Gestione del Database in Ambiente Cloud

La gestione cloud del database

Nel progetto **CoWorkSpace**, il database **PostgreSQL** può essere ospitato in cloud, cioè su infrastrutture remote gestite da provider come **Amazon Web Services (AWS)**, **Google Cloud Platform (GCP)** o **Microsoft Azure**.

Questo approccio permette di avere:

- Accesso da qualsiasi luogo
- Backup automatici
- Sicurezza avanzata
- Scalabilità per gestire più utenti e dati

Come funziona nel progetto

Il backend **Express** non si collega direttamente a un file o a un database locale.
Legge i parametri di connessione dal file .env, che contiene la connection string necessaria.

Path: backend/.env.example
Contiene le variabili di ambiente usate dal backend. La più importante è DATABASE_URL, che definisce la connessione a PostgreSQL. Questo file serve da modello: in cloud bisogna modificare host, user e password in base al provider.

Path: backend/src/config/database.js
Gestisce la connessione al database tramite **Sequelize**, usando DATABASE_URL. Configura anche l'opzione di logging delle query in base alla variabile REQUEST_LOGS. Qui viene creato l'oggetto sequelize usato da tutto il backend per le operazioni SQL.

Sicurezza in cloud

Quando il database è ospitato in cloud, è fondamentale proteggerlo da accessi non autorizzati.

Misure chiave:

- Accesso IP controllato (solo server autorizzati)
- Connessione cifrata con SSL/TLS
- Ruoli e permessi separati
- Rotazione periodica delle credenziali

Path: backend/src/middleware/auth.js
Middleware che gestisce l'autenticazione degli utenti. In futuro includerà la validazione di token JWT per controllare chi accede alle API. Questo livello di protezione si collega indirettamente al database, perché impedisce accessi non autorizzati agli endpoint che lo usano.

Path: backend/src/middleware/security.js
Middleware che gestisce la sicurezza delle richieste e il logging degli errori. Registra i problemi lato API e assicura che gli errori non blocchino il sistema. In caso di eccezioni, restituisce un errore standardizzato al client.

Backup e Ripristino

I provider cloud offrono backup automatici giornalieri, settimanali o personalizzati. Oltre a questi, il progetto include script manuali.

Path: backend/scripts/backup.sh
Script Bash che esegue un dump completo del database usando pg_dump. Può essere schedulato con cronjob o usato manualmente. È pensato come misura di backup aggiuntiva oltre a quella automatica del provider.

Path: backend/scripts/init_db.sql
Script SQL che crea le tabelle e i vincoli di base del database. Serve per inizializzare un nuovo database cloud alla prima installazione. È utile anche per ricostruire l'ambiente in caso di restore.

Path: backend/scripts/seed.js
Script che inserisce dati di esempio (utenti demo, spazi e prenotazioni). Aiuta nello sviluppo e nei test. Può essere usato anche in staging per popolare il database con dati realistici.

Monitoraggio e Performance

In cloud, è possibile monitorare in tempo reale:

- Connessioni attive
- Query lente
- Utilizzo di CPU e RAM
- Spazio disco

Ogni provider ha i suoi strumenti:

- AWS → CloudWatch
- GCP → Stackdriver
- Azure → Monitor

Path: backend/src/db.js
Inizializza la connessione a PostgreSQL usando Sequelize. Se REQUEST_LOGS=1, stampa in console le query eseguite. Questo file è il punto dove abilitare o disabilitare il logging SQL.

Path: backend/src/server.js
Avvia l'applicazione Express e mette in ascolto la porta configurata in .env. Qui si possono collegare strumenti di monitoraggio esterni, come Datadog o Prometheus. Inoltre, gestisce i log di avvio e cattura eventuali errori globali.

Gestione degli errori e rollback

Le operazioni critiche (come prenotazioni e pagamenti) usano transazioni SQL. Questo assicura che i dati vengano salvati solo se tutte le query hanno successo.

Path: backend/src/controllers/bookingsController.js
Qui vengono gestite le prenotazioni e i relativi pagamenti. Usa sequelize.transaction() per garantire l'integrità dei dati: se il pagamento fallisce, viene annullata anche la prenotazione. Questo approccio impedisce stati incoerenti nel database.

Deploy in cloud

Per creare il database in cloud si possono usare tre approcci:

- Caricare manualmente init_db.sql dal pannello del provider
- Eseguire lo script da linea di comando
- Automatizzare con strumenti come Terraform o CloudFormation

Path: backend/docker-compose.yml
File usato per avviare il backend e il DB in container Docker. È utile in sviluppo e può servire come base per la configurazione del deploy cloud. Contiene anche le variabili d'ambiente necessarie ai servizi.