

システムアーキテクト

Docker/Djangoを用いたWeb3層構造の実装

チームメンバー：渡邊諒（2442102）・田中誠真（2442054）

アーキテクチャ設計の基本戦略

環境の再現性: Dockerにより、「コード・ライブラリ・DB」をコンテナ化。どのPCでも簡単なコマンドで同一の開発環境を再現可能にしました。

堅牢なデータ管理: RDBMSとしてPostgreSQL 15を採用。ACID特性(原子性・一貫性)を重視し、業務データの整合性を保護しました。

セキュアなアクセス: ngrokを用いたトンネリングにより、HTTPSによる暗号化通信を実現しました。外部からの安全なアクセス経路を確保しています。

技術スタック

開発言語・フレームワーク:

Python 3.11 / Django 5.0: 迅速な開発が可能なMVTモデルを採用し、強力なORMと管理画面機能を活用しました。

データベース:

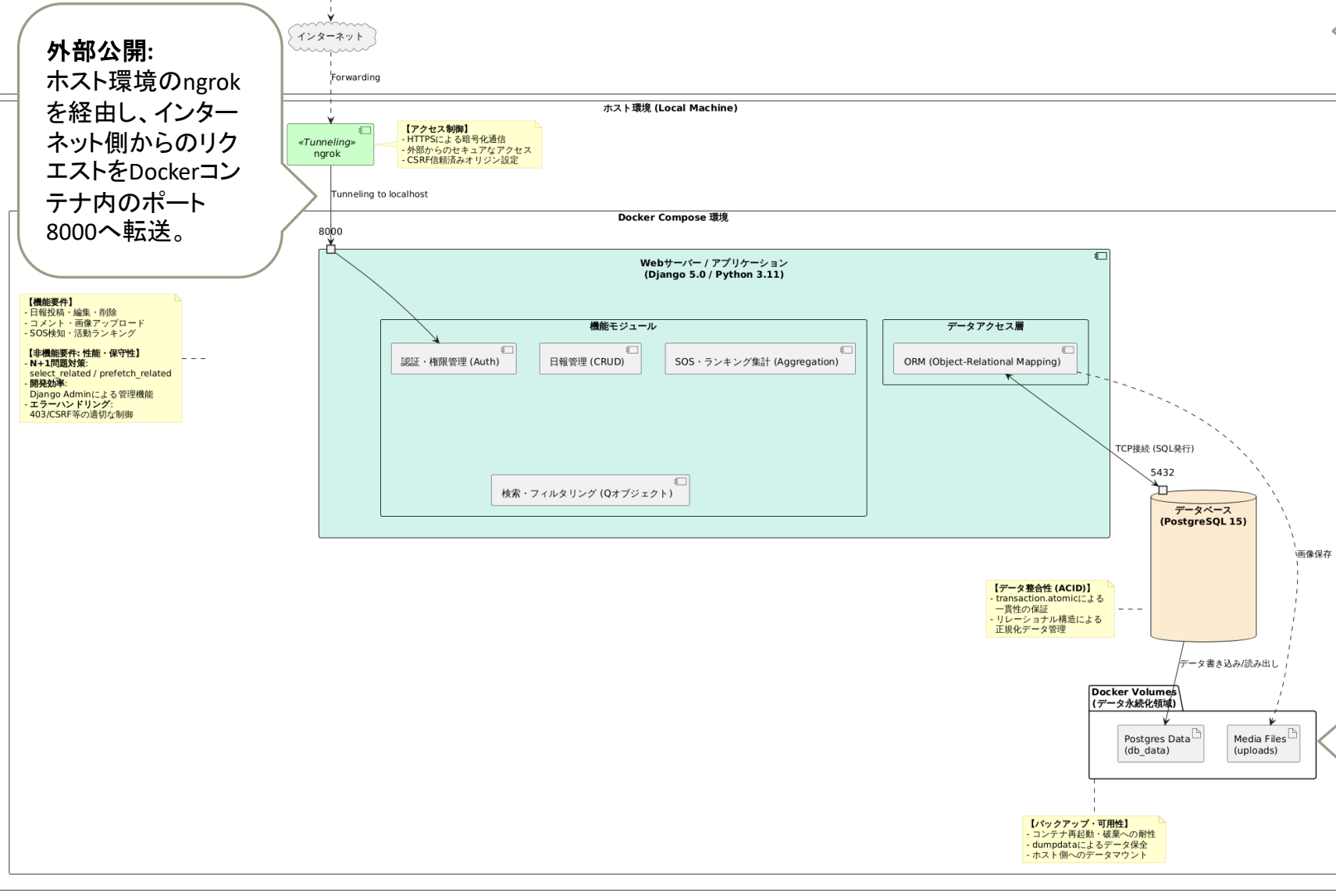
PostgreSQL 15: 複雑なクエリやトランザクション処理に強く、信頼性が高いリレーショナルデータベースです。

インフラ・フロントエンド:

Docker: 開発環境と本番環境の差異を極小化できます。

HTML5/CSS3: レスポンシブ対応により、PC・スマホ双方での利用を想定。

システム全体構成図



アプリケーション機能のポイント

認証・権限管理:

Django標準の認証システムを拡張し、安全なログイン管理を実現。

日報管理:

日報の作成・編集・削除機能に加え、画像アップロード機能も実装

アプリケーション機能のポイント

集計・分析ロジック:

- ・SOSアラート: 「調子」ステータスを監視し、不調なメンバーを抽出します。
- ・ランキング: 投稿数などを集計し、活動量を可視化できます。

検索・フィルタリング:

Qオブジェクト(Djangoのクエリビルダ)を用いて、キーワード検索とカテゴリ一絞り込みを組み合わせたAND/OR検索を実装しました。

DB設計と運用・保守

データアクセス層 (ORM):

- SQLを直接記述せず、Django ORMを介してDB操作を行うことで、SQLインジェクション等の脆弱性を排除。
- **N+1問題対策:** `select_related` / `prefetch_related` を使用し、クエリ発行回数を最適化してパフォーマンスを向上。