やることリスト
Type-2

メンバー

2271007 石井天斗

2171067 日高竜之介

GitHub repository

https://github.com/takato-0707/YarukotoList-Type-2

ユースケースと想定ユーザー

誰に使ってもらう？

・「規則的な活動」に価値を置くすべての人

どのような価値？

・「やったこと」を記録で確認、振り返ることで自己肯定感と満足感を得る

どのような記録を付ける？

・毎日の家事、仕事、趣味など

機能

タスク管理

・タスク作成・編集・削除: タイトル、説明、ジャンル、実行曜日を指定してタスクを管理
・曜日別割り当て: タスクごとに実行する曜日を複数選択可能（月～日）
・ジャンル分類: タスクをカテゴリー（ジャンル）で分類

進捗管理

・完了率表示: 当日割り当てられたタスクの完了率をプログレスバーで可視化
・完了記録: タスク完了時に日付とタイムスタンプを記録
・実行履歴: 過去30日間のタスク実行履歴を確認
・達成率統計: タスクごとの過去1ヶ月の達成率を表示

技術スタック

バックエンド
Flask: Webフレームワーク
SQLAlchemy: ORM
PostgreSQL: データベース
python-dotenv: 環境変数管理

フロントエンド
HTML/CSS3
Vanilla JavaScript: 非同期通信とDOM操作
Jinja2: テンプレートエンジン

データベース設計

曜日指定を独立テーブルに→ タスクと曜日の多対多関係を正規化

task_logは履歴テーブル→ タスクの定義（task）と実行結果（log）を分離

genreは任意紐付け→ タスクはジャンル未分類でも存在可能

UNIQUEで制約→実行記録が1日1回までに

genre: ジャンル管理

genre_id（主キー）

name（ジャンル名）

task_weekday: タスクの曜日割り当て

task_weekday_id（主キー）

task_id（外部キー）

weekday（曜日: 0=月曜, 6=日曜）

task: タスク管理

task_id（主キー）

title（タイトル）

description（説明）

genre_id（外部キー）

task_log: タスク実行記録

task_id（複合主キー）

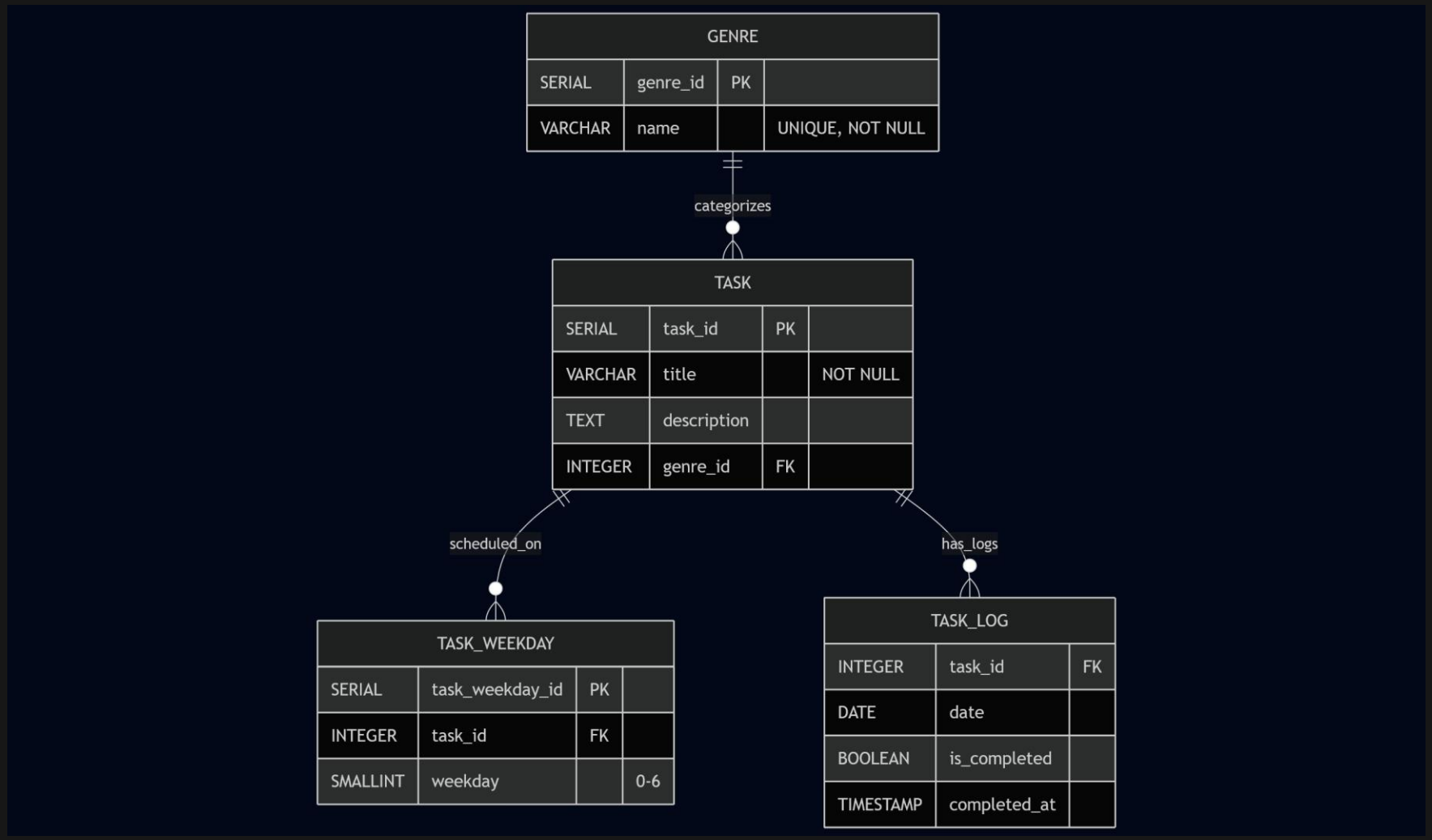
date（複合主キー）

is_completed（完了フラグ）

completed_at（完了日時）

ER図

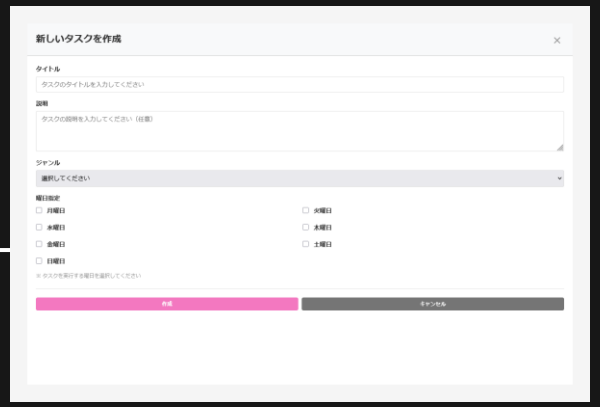
（Mermaid Live Editorで作成）

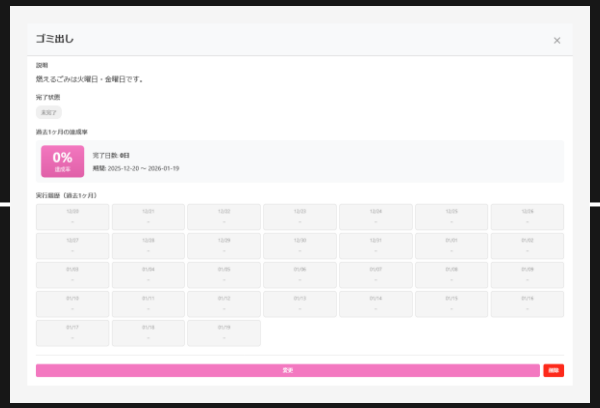


UI設計

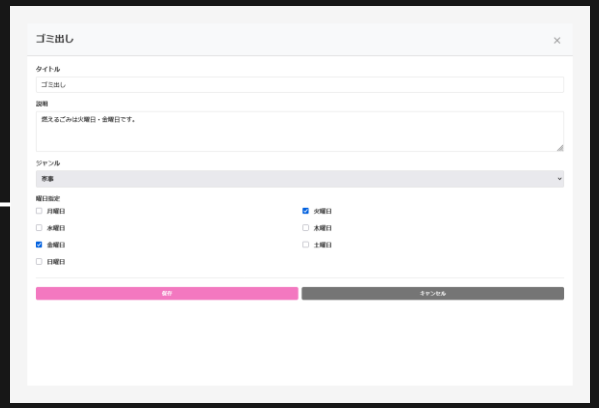
・極力1画面でまとめる、無駄な遷移はしない、
・基本操作はサイドバーで完結、タスク詳細、編集を右側メイン画面で
・非同期処理でタスク詳細を出す
・適切なハイライト、アニメーション

タスク作成





タスク詳細



タスク編集

アプリケーション設計 APIエンドポイント

ページ表示

GET / - メインページ（タスク一覧）

GET /task/<task_id> - タスク詳細

GET /task/create - タスク作成フォーム

データ操作

POST /task/create - タスク作成

POST /task/<task_id>/edit - タスク編集

POST /task/<task_id>/complete - タスク完了

POST /task/<task_id>/delete - タスク削除

プロジェクト構造

```
docker-flask-app-tx-v5.0/
├── app/
│   ├── __init__.py
│   ├── main.py             # アプリケーションエントリーポイント
│   ├── models.py           # データベースモデル
│   ├── requirements.txt    # Pythonパッケージ
│   ├── Dockerfile
│   ├── static/
│   │   └── css/
│   │       └── styles.css  # スタイルシート
│   └── templates/
│       ├── index.html      # メインページ
│       ├── create_memo.html # タスク作成フォーム
│       ├── create_task.html
│       └── view_memo.html  # タスク詳細ビュー
├── db/
│   ├── Dockerfile
│   └── init-sql/
│       ├── init-pgcrypto.sql
│       └── task_initialize.sql
├── compose.yml             # Docker Compose設定
└── .env                    # 環境変数（要作成）
```

動かし方

1,　.envファイルをルート直下に作る

```
# COMMON
API_PORT=3000
DB_PORT=5432
# PostgreSQL Settings
POSTGRES_USER=guest
POSTGRES_PASSWORD=password
POSTGRES_DB=guest
POSTGRES_HOST=db
```

.env

```
# COMMON
API_PORT=3000
DB_PORT=5432

# PostgreSQL Settings
POSTGRES_USER=guest
POSTGRES_PASSWORD=password
POSTGRES_DB=guest
POSTGRES_HOST=db
```

2,　コンテナを立ち上げる

docker compose up –d

3,　ページにアクセス

http://localhost:5000