

WEB プログラミングレポート

25G1053 齋藤翔太

2025 年 12 月 28 日

1. 開発者向け仕様書

1.1 開発環境とソースコード

本システムの開発環境とソースコードの構成について説明する。また，GitHub リポジトリの URL も記載する。

- リポジトリ URL：https://github.com/Lemon32528/webpro_06
- 開発環境：Node.js
- フレームワーク：Express
- テンプレートエンジン：EJS
- データ管理方式：データベースは使用せず，サーバプログラム内の配列変数としてメモリ上に保持する仕様とした。

1.2 各システムの概要

1.2.1 ポケモン対戦構築管理

1.2.1.1 データ構造

表 1.1 システム 1：データ構造（変数名：teamList）

変数名	型	説明
id	数値	一意の ID（連番）
name	文字列	チーム名（または構築名）
memo	文字列	補足・メモ

1.2.1.2 ページ遷移・機能詳細

表 1.2 システム 1：ページ遷移と機能詳細

メソッド	リソース名	機能・遷移内容
GET	/teams	一覧表示. 全データをテーブル形式で表示する.
GET	/teams/create	新規登録フォーム表示. public/teams_new.html へリダイレクト.
POST	/teams	新規登録処理. ID を付与して配列に追加し, 一覧へ戻る.
GET	/teams/:id	詳細表示. 指定 ID のデータの詳細を表示する.
GET	/teams/edit/:id	編集フォーム表示. 指定 ID のデータを初期値として表示.
POST	/teams/update/:id	更新処理. フォームの内容でデータを上書きし, 一覧へ戻る.
GET	/teams/delete/:id	削除処理. 指定データを削除し, 一覧へ戻る (確認ダイアログあり).

1.2.2 家計簿

1.2.2.1 データ構造

表 1.3 システム 2：データ構造 (変数名: kakeiboList)

変数名	型	説明
id	数値	一意の ID (連番)
date	文字列	日付 (YYYY-MM-DD 形式)
category	文字列	費目 (食費, 交通費など)
amount	数値	金額 (円)

1.2.2.2 ページ遷移・機能詳細

表 1.4 システム 2：ページ遷移と機能詳細

メソッド	リソース名	機能・遷移内容
GET	/kakeibos	一覧表示. 家計簿データを一覧表示する.
GET	/kakeibos/create	新規登録フォーム表示. public/kakeibos_new.html へリダイレクト.
POST	/kakeibos	新規登録処理. データを追加し, 一覧へ戻る.
GET	/kakeibos/:id	詳細表示. 指定 ID の詳細データを表示する.
GET	/kakeibos/edit/:id	編集フォーム表示.
POST	/kakeibos/update/:id	更新処理. データを更新し, 一覧へ戻る.
GET	/kakeibos/delete/:id	削除処理. データを削除し, 一覧へ戻る (確認ダイアログあり).

1.2.3 学食レビュー

1.2.3.1 データ構造

表 1.5 システム 3：データ構造 (変数名: reviewList)

変数名	型	説明
id	数値	一意の ID (連番)
menu	文字列	メニュー名
price	数値	価格 (円)
stars	数値	評価 (1~5 の整数)
comment	文字列	感想 (長文テキスト)

1.2.3.2 ページ遷移・機能詳細

表 1.6 システム 3：ページ遷移と機能詳細

メソッド	リソース名	機能・遷移内容
GET	/reviews	一覧表示. レビュー一覧を表示 (★マークで視覚化).
GET	/reviews/create	新規登録フォーム表示. public/reviews_new.html へリダイレクト.
POST	/reviews	新規登録処理. 必須入力チェックを行い, データを追加.
GET	/reviews/:number	詳細表示. 指定 ID の詳細データを表示する.
GET	/reviews/edit/:number	編集フォーム表示.
POST	/reviews/update/:number	更新処理. データを更新し, 一覧へ戻る.
GET	/reviews/delete/:number	削除処理. データを削除し, 一覧へ戻る (確認ダイアログあり).

1.3 採用した技術とその選定理由

本開発では, ユーザビリティの向上とデータの整合性を保つため, 授業で扱った範囲に加え, 以下の技術・仕様を採用した.

• required 属性 (HTML5)

- 概要: フォームの入力項目に付与することで, 未入力のまま送信されることをブラウザ側で防ぐ属性である.
- 採用理由: 家計簿の金額やレビューの評価など, システムにとって必須のデータが欠損したまま登録されるのを防ぐため, サーバ側の処理だけでなく, クライアント側でも入力を強制することで, 誤操作による空データの登録を未然に防止した.

• confirm メソッド (JavaScript)

- 概要: リンククリック時に「本当に削除してよろしいですか?」といった確認ダイアログを表示する機能である. onclick イベント内で使用し, キャンセルが選択された場合は遷移を中断する.
- 採用理由: データの削除処理において, 誤ってボタンを押してしまった場合に即座にデータが消失する事故を防ぐため, ユーザに再確認を促すステップを設けることで, 安全性・操作性を向上させた.

• textarea タグ (HTML)

- 概要: 複数行のテキスト入力が可能になるフォーム要素である.
- 採用理由: 「学食レビューの感想」や「ポケモン構築のコンセプト」など, 1 行のテキストボックス (input type="text") では視認性が悪く入力しづらい長文データを取り扱うため採用した. これにより, ユーザは改行を含む文章を容易に入力・確認できる.

- **Flexbox レイアウト (CSS)**

- **概要:** 要素の配置や順序, 間隔などを柔軟に設定できるレイアウトモジュールである.
- **採用理由:** 授業で扱った table タグによる画一的な配置だけでなく, トップメニューや一覧画面において, 画面幅に応じて要素が自動的に折り返されるカード型のレイアウトを実現するために採用した.

- **擬似クラス :hover (CSS)**

- **概要:** 要素が特定の状態 (マウスカーソルが乗っている等) にある場合にスタイルを適用する技術である.
- **採用理由:** ボタンやリンクにカーソルを合わせた際に色を変化させることで, クリック可能な要素であることを利用者に直感的に伝えるため (インタラクションの向上) に採用した.

- **white-space プロパティ (CSS)**

- **概要:** 要素内の空白や改行の扱いを指定するスタイル設定である.
- **採用理由:** 標準の HTML では無視されてしまう「入力フォーム内の改行」をそのまま画面に反映させ, 長文のデータ (詳細情報やレビューの感想など) を読みやすく表示するために採用した.

- **セマンティックタグ header, main (HTML)**

- **概要:** div タグのような意味を持たないタグではなく, ページの構造 (ヘッダー, メインコンテンツ) を明確にするタグである.
- **採用理由:** Web ページの文書構造を明確にし, 検索エンジンやブラウザがコンテンツの役割を正しく理解できるようにするため (アクセシビリティと保守性の向上) に採用した.

2. 管理者向け仕様書

本章では、本システムの管理者（運用担当者）に向け、システムの導入から運用，トラブルシューティングについて記述する。

2.1 インストールから起動までの手順確認

本システムを利用可能にするまでの大まかな手順は以下の通りである。各手順の詳細は後述する。

1. **環境確認:** Node.js がインストールされているか確認する。
2. **資材配置:** ソースコード一式をサーバ上の任意のディレクトリに配置する。
3. **依存ライブラリの導入:** `npm install` コマンドを実行する。
4. **起動:** `node app_repo.js` コマンドを実行し，ブラウザからアクセスする。

2.2 インストール方法

本システムは Node.js 環境上で動作する。以下の手順に従いインストールを行う。

2.2.1 事前準備

Node.js がインストールされていない場合は，公式サイトよりインストーラをダウンロードし，インストールを行うこと。

2.2.2 ライブラリのインストール

ターミナル（コマンドプロンプト）を開き，ソースコード（`app_repo.js` があるディレクトリ）へ移動した後，以下のコマンドを実行し，依存ライブラリをインストールする。

```
$ npm install
```

2.3 起動方法

インストールの完了後，以下のコマンドを実行してシステムを起動する。

```
$ node app_repo.js
```

コンソールに `Server listening...` と表示されれば起動成功である。Web ブラウザを立ち上げ，以下の URL へアクセスすることでシステムの利用を開始できる。

- アクセス URL: `http://localhost:8080`

2.4 起動できない場合の対処

コマンドを実行しても正常に起動しない場合、以下の原因と対処法を確認すること。

2.4.1 ポート番号が既に使用されている場合

エラー内容:

Error: listen EADDRINUSE: address already in use :::8080

対処法: ポート 8080 番が他のプロセス（以前起動したままのサーバ等）によって使用されている。

- 既に起動している node プロセスを終了させる。
- または、app.js 内のポート番号設定（8080）を別の番号に変更する。

2.4.2 モジュールが見つからない場合

エラー内容:

Error: Cannot find module 'express'

対処法: npm install が実行されていない、または失敗している可能性がある。再度 npm install を実行し、node_modules フォルダが生成されているか確認する。

2.5 終了方法

サーバを停止させる場合は、起動しているターミナル上でキーボードの Ctrl + C のキーを同時に入力する。

2.6 不具合（既知の制限事項）

本システムにおける現状の不具合、および仕様上の制限事項として**サーバ再起動によるデータ消失**が挙げられる。データベースを使用せず、プログラム内のメモリ（変数）上でデータを管理している仕様のため、サーバを停止・再起動すると、登録したデータはすべて初期状態にリセットされる。永続的なデータ保存が必要な場合は、データベースとの連携機能の実装が必要である。

3. 利用者向け仕様書

本章では，本システムに含まれる「ポケモン対戦パーティ管理システム」の利用手順について解説する．

3.1 システム概要

本アプリケーションは，ポケモン対戦で使用するパーティ（チーム）の構成を記録・管理するためのシステムである．パーティ名および構成する3匹のポケモンの詳細情報（努力値，技構成など）を一括して登録・保存し，ブラウザ上で閲覧することができる．

3.2 利用手順

3.2.1 システムの開始

Web ブラウザを起動し，以下の URL へアクセスする．

`http://localhost:8080`

アクセスするとトップページが表示される（図 3.1）．メニューから「ポケモンパーティ管理」のリンクをクリックすることで，パーティ一覧画面へ遷移する．なお，各システムの画面上部に表示されているヘッダーのタイトル（Web プログラミング課題）をクリックすると，いつでもこのトップメニュー画面に戻ることができる．



図 3.1 システムトップページ

3.2.2 パーティの確認と新規登録

一覧画面（図 3.2）では，登録済みのパーティ名称が一覧で表示される．新しいパーティを登録する場合は，画面内の「新規登録」リンクをクリックする．



図 3.2 パーティー一覧画面

「新規登録」をクリックすると，登録フォーム（図 3.3）が表示される．以下の項目を入力し，「登録」ボタンを押すことでデータが保存される．

- **パーティ名:** 構築の名称（例：S1 砂パ展開）
- **1 匹目～3 匹目:** 各ポケモンの名前と，詳細情報（性格・努力値・技構成など）



図 3.3 パーティ新規登録画面

3.2.3 詳細の確認・編集・削除

一覧画面にて、確認したいパーティのリンクをクリックすると、詳細画面（図 3.4）が表示され、登録されている 3 匹のすべての情報を閲覧できる。



図 3.4 パーティ詳細画面

詳細画面には以下の操作リンクが用意されている。

- **編集:** 登録内容を修正するためのフォームが開く。
- **削除:** このパーティ情報を削除し、一覧画面に戻る。

4. おわりに

本制作を通じて、Web アプリケーション開発におけるサーバサイドの挙動や、HTTP メソッド (GET/POST) によるデータの受け渡しについて深く理解することができた。特に、共通のテンプレート (ヘッダーや CSS) を利用して複数のシステム間でデザインを統一する作業は、保守性の高いコードを書くための良い実践となった。今回習得した Express や EJS の知識を基盤とし、今後はより複雑で規模の大きな Web システム開発にも挑戦していきたい。