

基本的なデータ表示の Web アプリ開発 仕様書

25G1XXX 氏名

2025 年 12 月 11 日

GitHub リポジトリ URL

本レポートで説明する 3 つの Web アプリケーションのソースコード一式は、次の GitHub リポジトリで管理している。

<https://github.com/jota92/wpro2025>

リポジトリ内には、以下の 3 つのディレクトリを用意し、それぞれ独立した Web アプリケーションとして実装した。

- `tasks_app` : 授業課題・テスト管理アプリ
- `subs_app` : サブスク管理アプリ
- `todos_app` : ToDo リストアプリ

いずれのアプリケーションも、授業で説明されたとおり、`Node.js`, `Express`, `EJS` を用いて実装し、各 `index.js` の先頭には `"use strict";` を付与している。授業で説明していない外部ライブラリやフレームワークは使用していない。

1 利用者向け仕様

この節では、実際に Web アプリケーションを利用するユーザの視点から、3 つのアプリケーションの目的と画面構成、および主な操作を説明する。

1.1 授業課題・テスト管理アプリ (`tasks_app`)

目的

大学の授業ごとに課題やテストが複数存在するため、「どの授業で、いつまでに、どの課題やテストがあるか」を一覧で把握できるようにすることを目的とする。本アプリケーションでは、授業

名, 課題名, 種別, 締切日, 状態を一元管理し, 締切の近い課題を見落とさないようにする.

画面構成

- **一覧画面 (トップ, /)** 登録済みの課題とテストを表形式で一覧表示する. 各行には「タイトル」「科目」「種別」「締切日」「状態」が表示され, 「詳細」リンクから詳細画面へ遷移できる. 画面上部には「新しい課題・テストを登録」リンクがあり, 新規登録フォームへ移動する.
- **詳細画面 (/tasks/:id)** 選択した 1 件の課題・テストについて, ID, タイトル, 科目, 種別, 締切日, 状態, メモを表示する. 「編集」リンクから編集フォームに遷移し, 「削除」ボタンを押すと削除処理を行う.
- **新規登録画面 (/create)** タイトル, 科目, 種別, 締切日, 状態, メモを入力し, 新しい課題・テストを登録するフォームである. 送信後は一覧画面へリダイレクトされ, 登録内容を確認できる.
- **編集画面 (/edit/:id)** 既存の課題・テストの内容を変更するフォームである. 送信後は当該課題の詳細画面へ遷移し, 変更結果を確認できる.

主な操作

利用者は次の操作を行う.

- 登録済みの課題・テストの一覧を確認する.
- 新しい課題・テストを登録する.
- 進捗に応じて状態 (未提出/勉強中/提出済) を更新する.
- 詳細画面でメモを追記し, 提出方法や注意事項を残す.
- 不要になった課題・テストを削除する.

1.2 サブスク管理アプリ (subs_app)

目的

動画配信サービスや音楽配信サービスなどのサブスクリプションが増えると, どのサービスにいくら支払っているか把握しづらくなる. 本アプリケーションは, 契約中のサブスクを一覧管理し, サービス名, カテゴリ, 料金, 課金サイクルなどを整理して確認できることを目的とする.

画面構成

- **一覧画面 (トップ, /)** 登録済みサブスクを表形式で一覧表示する. 各行には「サービス名」「カテゴリ」「料金」「課金サイクル」が表示され, 「詳細」リンクから詳細画面へ遷移できる. 画面上部のリンクから新規登録フォームへ移動する.
- **詳細画面 (/subs/:id)** 1 件のサブスクについて, ID, サービス名, カテゴリ, 料金, 課金サ

イクル、メモを表示する。「編集」リンクと「削除」ボタンを備える。

- **新規登録画面（/create）** サービス名、カテゴリ、月額料金、課金サイクル、メモを入力して登録するフォームである。
- **編集画面（/edit/:id）** 既存のサブスク情報を変更するフォームであり、料金変更やメモの追記などを行う。

主な操作

- 現在契約しているサブスク一覧を確認する。
- 新たに契約したサブスクを登録する。
- 料金変更や契約状況の変化に応じて内容を編集する。
- 解約したサブスクを削除して一覧を整理する。

1.3 ToDo リストアプリ（todos_app）

目的

日々の作業ややるべきことを簡単に整理できる ToDo リストを提供する。タスクのタイトル、優先度、完了状況などを一覧で管理し、重要なタスクを見落とさないようにすることを目的とする。

画面構成

- **一覧画面（トップ、/）** 登録済みの ToDo を表形式で一覧表示する。各行には「タイトル」「優先度」「完了フラグ」が表示され、「詳細」リンクから詳細画面へ遷移できる。画面上部のリンクから新規登録フォームへ移動する。
- **詳細画面（/todos/:id）** 1 件の ToDo について、ID、タイトル、優先度、完了フラグ、メモを表示する。編集と削除の操作を行うことができる。
- **新規登録画面（/create）** タイトル、優先度、完了フラグ（チェックボックス）、メモを入力して新しい ToDo を登録するフォームである。
- **編集画面（/edit/:id）** 既存の ToDo の内容を変更するフォームであり、タイトルの修正や完了状態の変更を行う。

主な操作

- 登録済み ToDo の一覧を確認する。
- 新しい ToDo を追加する。
- 進捗に応じて優先度や完了状態を更新する。
- 完了済みまたは不要な ToDo を削除する。

2 管理者向け仕様

この節では、データの構造や入力ルールなど、運用・データ管理の観点から 3 つのアプリケーションを説明する。いずれのアプリケーションも、データはサーバプログラム内の配列変数に保持し、データベース管理システムは用いていない。サーバを再起動すると配列は初期値に戻る。

2.1 授業課題・テスト管理アプリのデータ仕様

課題・テストは、表 1 に示す項目を持つオブジェクトとして管理する。

表 1 課題・テストのデータ項目

項目名	型	説明
id	数値	課題・テストを一意に識別する ID. 1 以上の整数。
title	文字列	課題またはテストのタイトル。
course	文字列	科目名（例：Web プログラミング）。
type	文字列	種別（レポート、テストなど）。
dueDate	文字列	締切日（例：2025-01-31）。
status	文字列	状態（未提出／勉強中／提出済）。
memo	文字列	任意のメモ。空文字列も可。

入力および運用上のルールは以下のとおりである。

- title, course, type, dueDate, status は必須項目とする。
- status は「未提出」「勉強中」「提出済」のいずれかを選択する。
- dueDate は HTML の日付入力欄を用い、YYYY-MM-DD 形式の文字列として扱う。
- 削除操作を行った課題は配列から完全に削除される。元のデータを復元する機能は持たない。

2.2 サブスク管理アプリのデータ仕様

サブスクリプション情報は、表 2 に示す項目を持つオブジェクトとして管理する。

運用ルールは以下のとおりである。

- name, category, fee, cycle は必須項目とする。
- fee は負の値を許可せず、フォーム上では 0 以上の整数のみ入力できるようにしている。
- サブスクを解約した場合は、削除処理で一覧から取り除く運用を想定する。

2.3 ToDo リストアプリのデータ仕様

ToDo データは、表 3 に示す項目を持つオブジェクトとして管理する。

表2 サブスクのデータ項目

項目名	型	説明
id	数値	サブスクを一意に識別する ID.
name	文字列	サービス名（例：Netflix）。
category	文字列	カテゴリ（動画配信／音楽／クラウドストレージなど）。
fee	数値	月額料金（円）。0 以上の整数を想定。
cycle	文字列	課金サイクル（毎月／毎年など）。
memo	文字列	備考・利用目的など。

表3 ToDo のデータ項目

項目名	型	説明
id	数値	ToDo を一意に識別する ID.
title	文字列	ToDo の内容を表すタイトル。
priority	文字列	優先度（高／中／低）。
done	真偽値	完了フラグ（true : 完了, false : 未完了）。
memo	文字列	補足メモ。

運用ルールは以下のとおりである。

- title は必須項目とし、空のまま登録することは想定しない。
- priority は「高」「中」「低」の中から選択させる。
- done はチェックボックスで入力し、チェック有無を真偽値に変換して保持する。

3 開発者向け仕様

この節では、実装者向けに技術的な仕様をまとめた。3つのアプリケーションはいずれも同様の構成とし、授業で扱った範囲の技術のみを用いて実装した。

3.1 共通の実装方針

- サーバサイドは Node.js と Express を用いる。
- テンプレートエンジンとして EJS を利用し、views ディレクトリ以下にテンプレートファイルを配置する。
- 静的ファイル(CSS)は public ディレクトリに配置し、app.use("/public", express.static(...)) で配信する。
- フォームから送信されるデータの取得には express.urlencoded({ extended: true }) を用いる。

- 各アプリケーションの `index.js` の先頭に`"use strict";` を記述する。
- データはサーバプログラム内の配列変数 (`tasks`, `subscriptions`, `todos`) として保持し、授業で扱っていないデータベースは使用しない。

3.2 授業課題・テスト管理アプリのエンドポイント

授業課題・テスト管理アプリ (`tasks_app`) の主なエンドポイントを表 4 に示す。

表 4 授業課題・テスト管理アプリのエンドポイント

メソッド	パス	説明
GET	/	課題・テストの一覧を表示する。
GET	/tasks/:id	指定 ID の課題・テストの詳細を表示する。
GET	/create	新規登録フォームを表示する。
POST	/create	フォーム入力から新しい課題・テストを作成し配列に追加する。
GET	/edit/:id	指定 ID の課題・テストの編集フォームを表示する。
POST	/edit/:id	編集内容で既存の課題・テストを更新する。
POST	/delete/:id	指定 ID の課題・テストを配列から削除する。

一覧や詳細表示には HTTP GET を用い、登録・更新・削除には HTTP POST を用いるという基本的な操作を統一している。

3.3 サブスク管理アプリのエンドポイント

サブスク管理アプリ (`subs_app`) の主なエンドポイントを表 5 に示す。

表 5 サブスク管理アプリのエンドポイント

メソッド	パス	説明
GET	/	サブスク一覧を表示する。
GET	/subs/:id	指定 ID のサブスク詳細を表示する。
GET	/create	新規登録フォームを表示する。
POST	/create	フォーム入力からサブスクを作成し配列に追加する。
GET	/edit/:id	指定 ID のサブスク編集フォームを表示する。
POST	/edit/:id	編集内容で既存のサブスクを更新する。
POST	/delete/:id	指定 ID のサブスクを配列から削除する。

3.4 ToDo リストアプリのエンドポイント

ToDo リストアプリ (`todos_app`) の主なエンドポイントを表 6 に示す。

表6 ToDo リストアプリのエンドポイント

メソッド	パス	説明
GET	/	ToDo の一覧を表示する.
GET	/todos/:id	指定 ID の ToDo 詳細を表示する.
GET	/create	新規登録フォームを表示する.
POST	/create	フォーム入力から ToDo を作成し配列に追加する.
GET	/edit/:id	指定 ID の ToDo 編集フォームを表示する.
POST	/edit/:id	編集内容で既存の ToDo を更新する.
POST	/delete/:id	指定 ID の ToDo を配列から削除する.

3.5 動作環境と起動方法

3つのアプリケーションはいずれも、Node.js がインストールされた環境で次の手順により起動する。例として `tasks_app` の場合を示す。

1. リポジトリを取得し、`tasks_app` ディレクトリに移動する。
2. 一度だけ `npm install` を実行して依存パッケージをインストールする。
3. `npm start` または `node index.js` を実行し、8080 番ポートでサーバを起動する。
4. ブラウザから `http://localhost:8080/` にアクセスすると、各アプリケーションのトップページが表示される。

サブスク管理アプリ、ToDo リストアプリについても、それぞれ `subs_app`、`todos_app` ディレクトリで同様の手順を実行することで起動できる。