# 『タスク管理アプリ仕様書』

24G1099 寺田幸正

2025 年 1 月 6 日

## 1　利用者向け仕様

### 1.1　アプリ概要

このタスク管理アプリでは, ユーザーがタスクを追加, 削除, 状態の切り替えを行うことができる. また登録されたタスクを一覧で確認することができる.

### 1.2　主な機能

**1.　タスク追加**

ユーザーはタスクを入力し,「タスク追加」ボタンをクリックすることで新しいタスクを追加することができる.

**2.　タスク一覧表示**

「タスク一覧」ボタンをクリックすると, 登録されたタスクが一覧表示される. 各タスクは完了状態 (チェックボックス) や削除ボタンを含んでいる.

**3.　タスク削除**

一覧からタスクを削除することができる. タスクの横にある「削除」ボタンをクリックすると, そのタスクが削除される.

**4.　タスク完了・未完了の切り替え**

各タスクにチェックボックスがあり, チェックを入れることでタスクの完了状態を変更できる. 完了したタスクは取り消し線が引かれ, 文字の色がグレーに変わる.

### 1.3　インターフェース

**1.　タスク入力フォーム**

入力欄：タスク名を入力するためのテキストボックス

送信ボタン：タスクを追加するためのボタン

**2.　タスク一覧ボタン**

タスクの一覧を表示するためのボタン

**3.　タスク表示エリア**

タスクの内容, 完了状態 (チェックボックス), 削除ボタンが表示される.

## 1.4　画面遷移

**1.　タスク追加**

タスク入力後,「タスク追加」ボタンをクリックすると, タスクがサーバーに送信され追加が完了する.

**2.　タスク一覧表示**

「タスク一覧」ボタンをクリックすると, サーバーからタスク一覧が取得され, 画面に表示される.

## 1.5　使用手順

図 1 と図 2 にタスク管理アプリの使用手順をフローチャートとして示している.

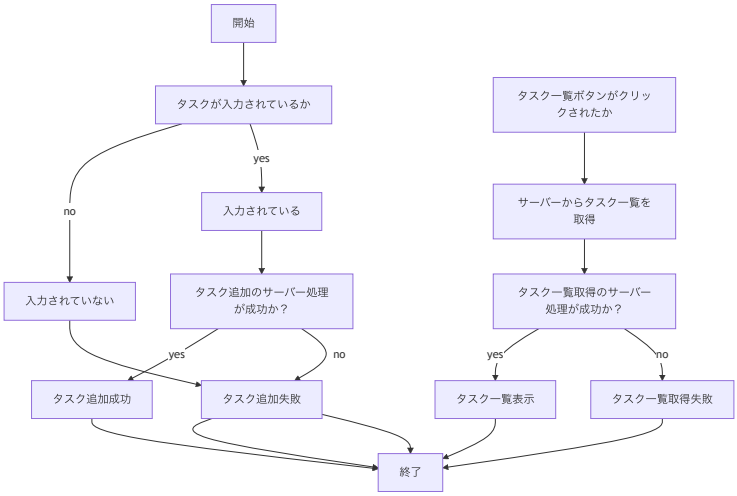


図 1　タスク追加・一覧取得のフローチャート.

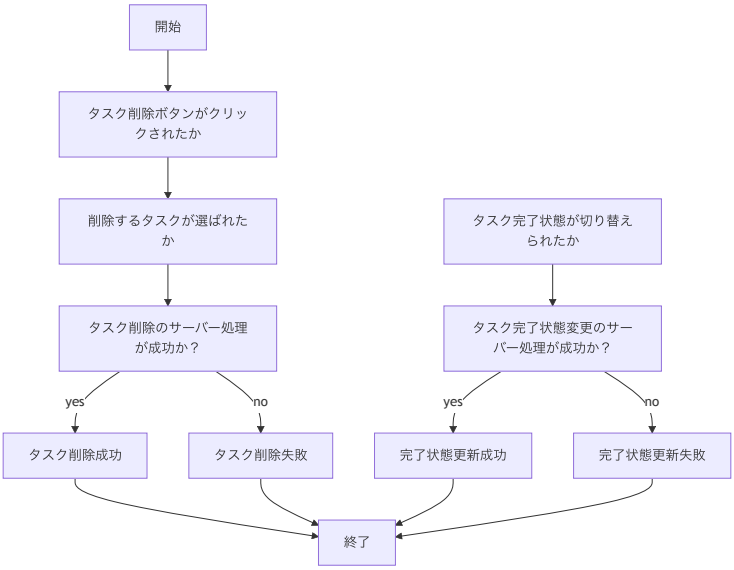


図 2　タスク削除・完了状態変更のフローチャート.

# 2　管理者向け仕様

## 2.1　起動方法

1. まずはじめにサーバを起動する. サーバは `node app25.js` のコマンドを実行して起動させる.
2. 次にサーバが起動したら,Web ブラウザで `http://localhost:8080/bbs2.html` にアクセスする.

## 2.2　サーバサイドでの処理

**1.　タスクの追加・削除**サーバーは POST リクエストを受け取ると, タスクを配列に追加または削除する. データは JSON 形式で返され, エラーハンドリングを行う.

**2.　タスク完了状態の更新**タスクが完了した場合, チェックボックスの状態が変更され, タスクの状態がサーバーに送信される.

# 3　開発者向け仕様

## 3.1　ファイル一覧

表 1　ファイル一覧

| ファイル名 | 役割 | 主な内容 |
|---|---|---|
| `bbs2.html` | ユーザーインターフェースを提供 | タスク入力フォーム, 一覧表示ボタン, 表示エリアを定義 |
| `bbs2.css` | アプリケーションのスタイルを定義 | タスクの完了状態に応じたスタイル, 項目のデザイン, 削除ボタンのスタイル |
| `bbs2.js` | クライアントサイドの動作を管理 | タスクの追加, 一覧取得, 削除, 完了状態の切り替えの処理 |
| `app25.js` | サーバサイドの処理 | タスクの追加, 取得, 削除, 完了状態の変更を行う API を実装 |

## 3.2　採用技術の概要と採用理由

### 3.2.1　タスクの削除方法

タスクの削除は, サーバーに対して POST メソッドを使用してリクエストを送信することで行う. サーバー側では, 指定されたタスク ID を基にタスクを削除し, その後更新されたタスク一覧を返す.

**技術:　POST メソッド + タスク ID の指定**

**・概要:** タスクを削除するために, クライアントは削除対象のタスク ID をサーバーに POST リクエストとして送信する. サーバーはその ID に基づいてタスクを配列から削除し, 更新されたタスク一覧を返す.

**・採用理由:** サーバー側でタスクの削除だけでなく, 削除後のタスク一覧の取得や関連するデータの更新を同時に行うことができ柔軟性に優れているためである.

### 3.2.2　タスク完了状態変更

タスクの完了状態の変更は, タスクが完了したかどうかを管理するために必要な操作である. タスクが完了した場合は, その完了状態をサーバー側で更新し, クライアント側で完了状態を反映させる.

**技術:　POST メソッド + タスク ID + 完了状態**

**・概要:** タスクの完了状態を変更するために, クライアントからタスク ID と完了状態 (true または false) を POST メソッドでサーバーに送信する. サーバー側では, そのタスクの完了状態を変更し, 更新されたタスク一覧を返す.

**・採用理由:** クライアント側で「完了」ボタンを押すだけでサーバーと通信し, 状態変更が即座に反映されるため, ユーザー体験が向上するからである.

## 3.3　内部的な動作

**1.　タスク追加**

新しいを取得し, `tasks` 配列に追加する. その後更新された `tasks` 配列を返す.

**2.　タスク削除**

指定された `taskId` を使って, `tasks` 配列内からタスクを削除する. `taskId` が存在しない場合, エラーレスポンスをを返す. その後, 削除後の `tasks` 配列を返す.

**3.　タスク完了状態変更**

指定された `taskId` のタスクの完了状態 ( `completed` ) を反転させる. `taskId` が存在しない場合, エラーレスポンスをを返す. その後, 更新後の `tasks` 配列を返す.

**4.　タスク一覧取得**

現在のタスク一覧 ( `tasks` ) 配列を返す.

表 2　プログラムの説明

| 操作 | パラメータ | 返り値 | 説明 |
|---|---|---|---|
| タスク一覧 | `taskName, completed` | `tasks` | 新しいタスクを作成して, `tasks` 配列に追加する |
| タスク削除 | `taskId` | `tasks, error` | 指定した `taskId` を持つタスクを削除し, 更新後の `tasks` 配列を返す |
| タスク完了状態変更 | `taskId` | `tasks, error` | 指定した `taskId` のタスクの完了状態を反転し, 更新後の `tasks` 配列を返す |
| タスク一覧取得 | なし | `tasks` | 現在のすべてのタスクの一覧を返す |

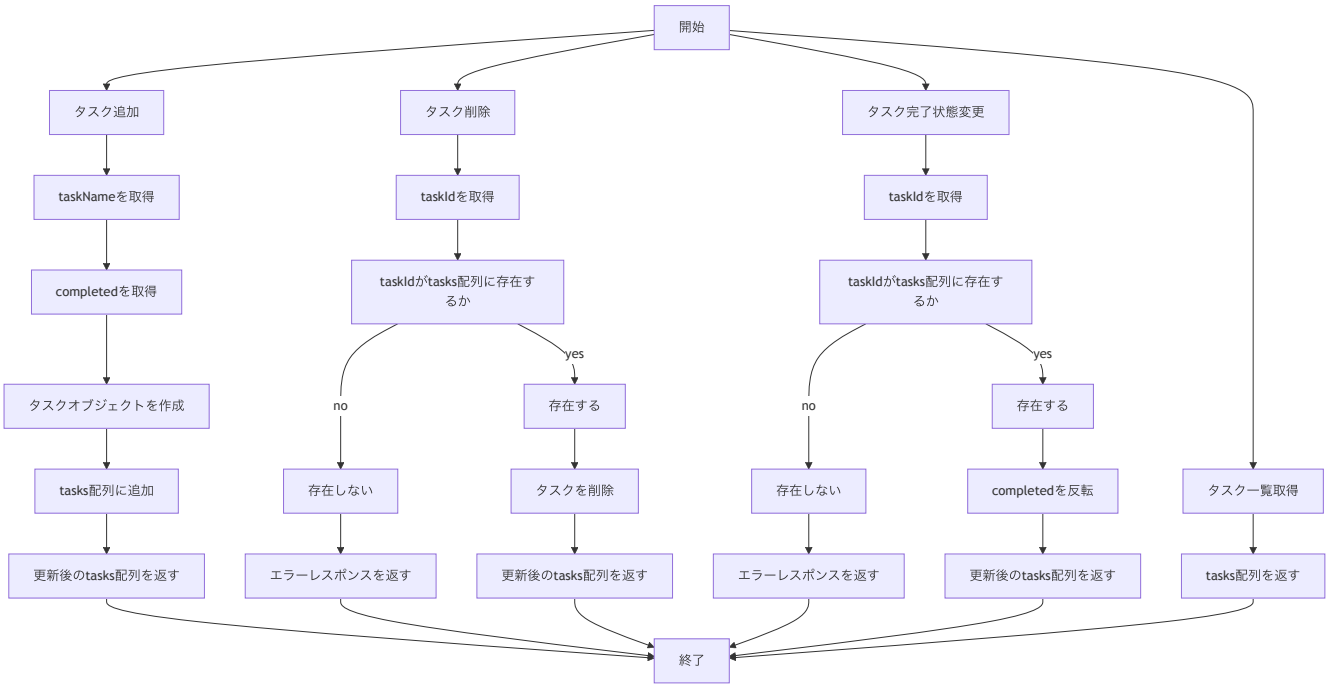


図 3　プログラム全体のフローチャート.

# 4　付録

`http://localhost:8080/bbs2.html`

`https://github.com/Yukimasa1126/webpro_06`