一覧表示システム
Webアプリケーション仕様書

25G1034 恩田隼士

2025年12月26日

目 次

ソースコード

本仕様書で使用したソースコードを添付する.
https://github.com/crab2424/webpro_submit

第Ⅰ部

利用者向け仕様書

第1章　利用者向け仕様書

1.1　概要

このシステムは、日本の都道府県データを Web ブラウザ上で閲覧・管理できるアプリケーションである。一覧表示、詳細情報の確認に加え、新しい都道府県データの追加、既存データの編集および削除を行うことができる。

1.2　使用できる機能

このシステムでは以下の機能を使用することができる。

- 都道府県データの一覧表示
- 各都道府県の詳細情報（人口、面積、県庁所在地など）の表示
- 新規データの登録
- データの編集
- データの削除

1.3　画面構成と操作方法

1.3.1　起動画面（トップページ）

システムにアクセスすると、各管理システムへのリンクが表示されるトップページが開く。「都道府県一覧表示システム」のリンクをクリックすることで、一覧画面へ遷移する。

図 1.1: トップページ

1.3.2 一覧表示

登録されている都道府県データが表形式で表示される。

- 詳細リンク: 都道府県名をクリックすると、詳細画面へ移動する。
- 追加ボタン: 画面上部のボタンから新規作成画面へ移動する。

図 1.2: 一覧表示

1.3.3 詳細表示

選択した都道府県の詳しい情報（コード、面積、人口、県庁所在地、地方区分）を確認できる。この画面から、情報の「編集」または「削除」を行う画面へ移動できる。

図 1.3: 詳細表示

1.3.4　データ追加

新規作成画面にて、必要な情報を入力し「登録」ボタンを押すことで、新しいデータをリストに追加できる。「登録して新たに作成」を選択すると、連続してデータを入力することが可能である。

1.3.5　データ編集

詳細画面の「編集」ボタンから、既存のデータを修正できる。内容は即座に一覧および詳細画面に反映される。

1.3.6　データ削除

詳細画面の「削除」ボタンを押すと、確認画面が表示される。削除対象のデータを確認し、問題なければ実行することでデータが削除される。

1.4　Q&A (よくある質問)

Q. 登録したデータが消えてしまった

A. システムの仕様上、サーバーを停止または再起動すると、登録・編集した内容はすべて初期状態（インストール時のデータ）に戻ります。

Q. スマートフォンから利用できるか？

A. Web ブラウザ（Chrome, Safari 等）が搭載された端末であれば、PC と同様に利用可能です。

第II部

管理者向け仕様書

第2章　管理者向け仕様書

2.1　概要

本仕様書は，Node.js およびテンプレートエンジン EJS を用いた Web アプリケーションの仕様書である．本システムは，サーバーサイドで都道府県データを管理し，EJS を用いて動的に HTML を生成・表示する．データベースの利用は行わず，サーバープロセスのメモリ上（変数）でデータを保持・操作することを前提とする．なお，本システムは macOS 環境で開発したものであるため，macOS を基準とした管理方法を示す．

2.2　インストール方法

2.2.1　動作環境

本システムの動作環境およびインストール手順を以下に示す．

- OS: Windows, macOS, または Linux
- サーバー実行環境: node.js (開発時バージョン v24.12.0)
- パッケージマネージャー: npm, nodebrew, homebrew

2.2.2　セットアップ手順

homebrew のインストール

JavaScript を動作させるにあたって，macOS では homebrew を用いて nodebrew をインストールする必要がある．まず，ターミナル上で以下のコマンドを実行する．その際，ターミナル上でパスワードを求められるので，パスワードを入力する．

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEA
```

次に，以下の 2 つのコマンドを順に実行する．

```
( echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)"') >> ~/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"
```

nodebrew のインストール

ターミナルにて，以下のコマンドを 4 つを 1 つずつ順に実行して nodebrew をインストールする．

```
$ brew install nodebrew
$ nodebrew setup
$ echo 'export PATH=$HOME/.nodebrew/current/bin:$PATH' >> ~/.zshrc
$ source ~/.zshrc
```

node.js のインストール

ターミナルにて，以下のコマンドを 1 つずつ順に実行して node.js をインストールする．

```
$ nodebrew install stable
$ nodebrew ls
```

これにより，現在のバージョンが表示されるため，それに従って以下のコマンドを実行する．

```
$ nodebrew use v24.12.0
$ npm install -g npm
```

2.3 起動・終了方法

2.3.1 ソースコード

以下の URL からソースコードをダウンロードすることができる．`https://github.com/crab2424/webpro_submit`

2.3.2 起動方法

まず，ソースコードが配置されたディレクトリ（webpro_submit/app）に移動する．

```
$ cd webpro_submit/app
```

次に，以下のコマンドを実行してサーバーを起動する．

```
$ node app_system.js
```

起動に成功すると，コンソールに以下のメッセージが表示される．

```
Example app listening on port 8080!
```

その後，Web ブラウザで以下の URL にアクセスすると，トップページが表示される．

`http://localhost:8080/`

2.3.3 終了方法

サーバーを実行しているターミナルにおいて，以下のキーを入力してプロセスを終了する．

Ctrl + C

2.4 トラブルシューティング

2.4.1 起動できない場合

- エラー: Address already in use
  ポート 8080 が他のプロセスで使用されている可能性がある．他の Node.js プロセスを終了するか，PC を再起動してから再度実行すること．

- エラー: Cannot find module
  npm install が正しく実行されていない可能性がある．再度インストール手順を確認すること．

2.5 既知の不具合・制限事項

本システムは学習用アプリケーションとしての仕様上, 以下の制限事項が存在する．

- データの非永続性: データベースを使用せず，サーバーのメモリ上（変数）でデータを管理しているため，サーバーを再起動または終了すると，追加・編集・削除したデータは初期状態にリセットされる．

第III部

開発者向け仕様書

第3章　開発者向け仕様書：都道府県一覧表示システム

3.1　概要

本仕様書は，Node.js およびテンプレートエンジン EJS を用いた「都道府県一覧表示システム」の設計仕様書である．本システムは，サーバーサイドで都道府県データを管理し，EJS を用いて動的に HTML を生成・表示する．データベースの利用は行わず，サーバープロセスのメモリ上（変数）でデータを保持・操作することを前提とする．

3.2　データ管理

3.2.1　データ構造

サーバー内の配列変数で管理するデータ構造は表 3.1 の通りに構成する．

表 3.1: 都道府県データ構造

プロパティ名	データ型	説明
id	Number	一意な識別子
name	String	都道府県名
code	Number	都道府県番号
area	Number	面積（km^2）
population	Number	人口（人）
capital	String	県庁所在地
region	String	地方区分

id を除くすべてのデータは入力フォームで作成，編集が可能である．id はデータ配列の長さに基づいて自動採番するため，作成時に入力不要である．

3.2.2　表示における制約事項

本システムでは，データ id と配列の整合性を保つため，以下の仕様を採用する．

- id と配列: 配列と id の扱いを簡易化させるため，配列の 0 番目にはダミーデータを作成し，システム上では非表示とする．
- 削除処理: データを削除する際は，配列から要素を取り除くのではなく，該当のデータを null に置き換える．これにより，他のデータの配列と id がずれることを防ぐ．
- 一覧表示の制御: EJS テンプレートの forEach ループにおいて，要素がダミーデータまたは null でない場合のみ行を描画する条件分岐（if 文）を設ける．

また，画面のレイアウトおよび配色は，style.css に定義したスタイルを用いる．css は，3 つのシステムでレイアウトを統一するため，1 つのファイルで管理する．

3.3 ディレクトリ構成

本システムは，図 3.1 に示すディレクトリ構造に従って，使用するファイルを配置する．

```
webpro_submit/
    |- app/
        |- app_system.js            (メインロジック・データ変数保持)
        |- public/                  (静的ファイル)
            |- style.css            (CSS ファイル)
            |- pref_new.html        (新規作成フォーム)
        |- views/                   (EJS テンプレート)
            |- landing.ejs          (トップページ)
            |- pref/                (都道府県システム)
                |- pref_check.ejs   (削除確認画面)
                |- pref_detail.ejs  (詳細表示画面)
                |- pref_edit.ejs    (編集フォーム)
                |- pref.ejs         (一覧表示画面)
```

図 3.1: ディレクトリ構成

3.4 HTTP メソッドとルーティング

本システムにおける各 URL と HTTP メソッド，および対応する処理を表 3.2 に定義する．

表 3.2: ルーティング一覧

機能	メソッド	パス (URL)	対応ビュー
トップページ	GET	/	views/landing.ejs
一覧表示	GET	/pref	views/pref/pref.ejs
新規作成フォーム	GET	/pref/create	public/pref_new.html
詳細表示	GET	/pref/:id	views/pref_detail.ejs
編集フォーム	GET	/pref/edit/:id	views/pref_edit.ejs
削除確認	GET	/pref/check/:id	views/pref_check.ejs
新規データ作成	POST	/pref	処理後一覧を表示
新規データ作成	POST	/pref/create	処理後新規作成へリダイレクト
データ更新	POST	/pref/update/:id	処理後詳細ページを表示
データ削除	GET	/pref/delete/:id	処理後一覧へリダイレクト

3.5 ページ遷移

本システムにおける画面間の遷移を図 3.2 に示す．なお，本システムのページには戻るリンクを配置するため，一覧表示ページ及び詳細表示ページに直接遷移することが可能である．

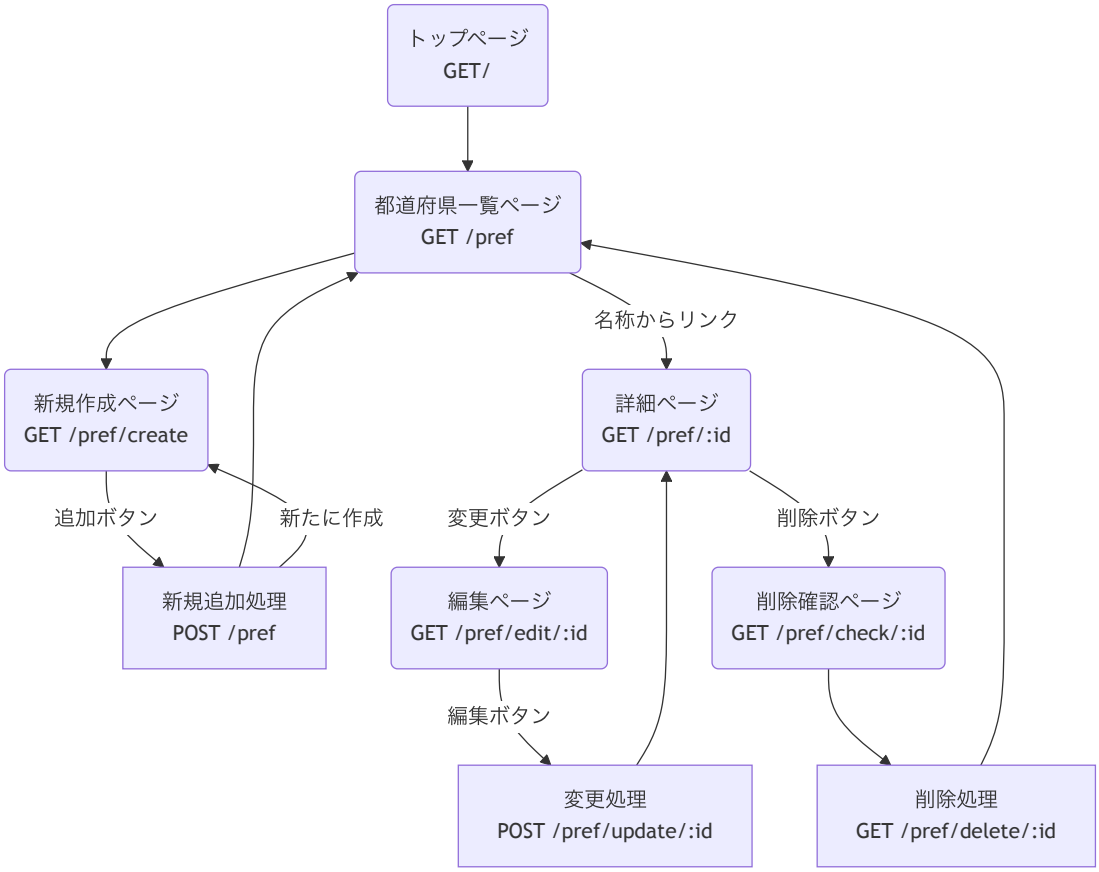

図 3.2: 画面遷移フローチャート

3.6 各機能・リソース詳細

3.6.1 トップページ

- URL: /
- 処理: views/landing.ejs を表示する.
- 要素: 各システムへのリンク

3.6.2 一覧表示機能

- URL: /pref
- 処理: サーバー変数の全データを EJS に渡し，forEach 文を用いてループ処理を行う.

  - 条件付きレンダリング: 本システムでは id 整合性のために空データ (null) やダミーデータ (id:0) を保持している．そのため，forEach ループ内で条件分岐を行い，有効なデータ（null かつ id:0 でないもの）のみを HTML テーブルとして出力する．

- 要素: 各行ごとの要素の名前に対応する詳細リンク，追加ボタン

3.6.3 新規作成機能

- フォーム: GET /pref/create
  - 処理: 表 3.1 の id を除くすべてのプロパティを入力フィールドとして表示する．
  - 送信先: POST /pref
  - 要素: 各プロパティに対応する入力フィールド，登録ボタン，登録後新規作成ボタン
- 作成処理: POST /pref
  - リクエストボディから値を取得．
  - 新しい id を採番し，サーバー変数（配列）に push する．
  - 新規作成したデータの内容をサーバーのターミナルに出力する．
  - 処理完了後，一覧画面 (/pref) へリダイレクトする．

3.6.4 詳細表示機能

- URL: /pref/:id
- 処理: URL パラメータの id に基づき配列を検索し，対象データを表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/pref) へリダイレクトする．
- 要素: 編集ボタン，削除ボタン，一覧に戻るリンク

3.6.5 編集・更新機能

- フォーム: GET /pref/edit/:id
  - 処理: 対象データを検索し，value 属性に現在の値を埋め込んで表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/pref) へリダイレクトする．

  - 送信先: POST /pref/update/:id

- 更新処理: POST /pref/update/:id
  - id に基づき配列内の該当要素を特定.
  - リクエストボディの値でプロパティを上書きする.
  - 表示: 更新内容をサーバーのターミナルに出力する.
  - 更新後，詳細画面 (/pref/:id) を表示する.

3.6.6 削除機能

- フォーム: GET /pref/check/:id
  - 簡易確認フォームを表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/pref) へリダイレクトする.
  - 確認後、GET /pref/delete/:id で削除処理を実行する.

- 削除処理: GET /pref/delete/:id
  - id に対応する配列要素に null を代入し，データを削除状態にする（配列の要素数は変更しない）．削除対象の指定した id に対応するデータが元から存在しない場合は，一覧表示ページ (/pref) へリダイレクトする.
  - 削除した要素名をコンソールに出力する.
  - 削除完了後，一覧画面 (/pref) へリダイレクトする.

第4章　開発者向け仕様書：星座一覧表示システム

4.1　概要

本仕様書は，Node.js およびテンプレートエンジン EJS を用いた「星座一覧表示システム」の設計仕様書である．本システムは，サーバーサイドで星座データを管理し，EJS を用いて動的に HTML を生成・表示する．データベースの利用は行わず，サーバープロセスのメモリ上（変数）でデータを保持・操作することを前提とする．

4.2　データ管理

4.2.1　データ構造

サーバー内の配列変数で管理するデータ構造は表 4.1 の通りに構成する．

表 4.1: 星座データ構造

プロパティ名	データ型	説明
id	Number	一意な識別子
name	String	星座名
en	String	英語表記
shape	String	星座の形
height	String	高度 (°)
star	String	代表する星
season	String	季節

id を除くすべてのデータは入力フォームで作成，編集が可能である．id はデータ配列の長さに基づいて自動採番するため，作成時に入力不要である．

4.2.2　表示における制約事項

本システムでは，データ id と配列の整合性を保つため，以下の仕様を採用する．

- id と配列: 配列と id の扱いを簡易化させるため，配列の 0 番目にはダミーデータを作成し，システム上では非表示とする．
- 削除処理: データを削除する際は，配列から要素を取り除くのではなく，該当のデータを null に置き換える．これにより，他のデータの配列と id がずれることを防ぐ．
- 一覧表示の制御: EJS テンプレートの forEach ループにおいて，要素がダミーデータまたは null でない場合のみ行を描画する条件分岐（if 文）を設ける．

また，画面のレイアウトおよび配色は，style.css に定義したスタイルを用いる．css は，3 つのシステムでレイアウトを統一するため，1 つのファイルで管理する．

4.3 ディレクトリ構成

本システムは，図 4.1 に示すディレクトリ構造に従って，使用するファイルを配置する．

```
webpro_submit/
    |- app/
        |- app_system.js              (メインロジック・データ変数保持)
        |- public/                    (静的ファイル)
            |- style.css              (CSS ファイル)
            |- stella_new.html        (新規作成フォーム)
        |- views/                     (EJS テンプレート)
            |- landing.ejs            (トップページ)
            |- stella/                (星座システム)
                |- stella_check.ejs   (削除確認画面)
                |- stella_detail.ejs  (詳細表示画面)
                |- stella_edit.ejs    (編集フォーム)
                |- stella.ejs         (一覧表示画面)
```

図 4.1: ディレクトリ構成

4.4 HTTP メソッドとルーティング

本システムにおける各 URL と HTTP メソッド，および対応する処理を表 4.2 に定義する．

表 4.2: ルーティング一覧

機能	メソッド	パス (URL)	対応ビュー
トップページ	GET	/	views/landing.ejs
一覧表示	GET	/stella	views/stella/stella.ejs
新規作成フォーム	GET	/stella/create	public/stella_new.html
詳細表示	GET	/stella/:id	views/stella_detail.ejs
編集フォーム	GET	/stella/edit/:id	views/stella_edit.ejs
削除確認	GET	/stella/check/:id	views/stella_check.ejs
新規データ作成	POST	/stella	処理後一覧を表示
新規データ作成	POST	/stella/create	処理後新規作成へリダイレクト
データ更新	POST	/stella/update/:id	処理後詳細ページを表示
データ削除	GET	/stella/delete/:id	処理後一覧へリダイレクト

4.5 ページ遷移

本システムにおける画面間の遷移を図 4.2 に示す．なお，本システムのページには戻るリンクを配置するため，一覧表示ページ及び詳細表示ページに直接遷移することが可能である．

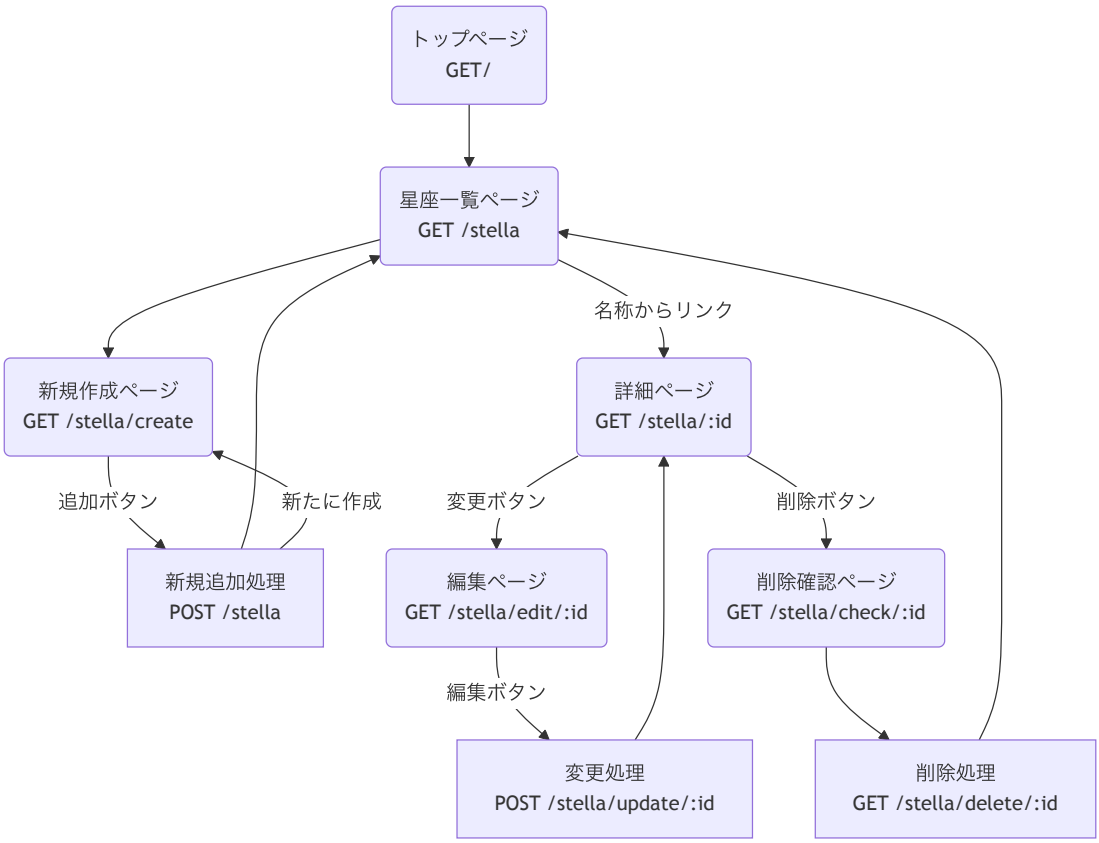

図 4.2: 画面遷移フローチャート

4.6 各機能・リソース詳細

4.6.1 トップページ

- URL: /
- 処理: views/landing.ejs を表示する.
- 要素: 各システムへのリンク

4.6.2 一覧表示機能

- URL: /stella
- 処理: サーバー変数の全データを EJS に渡し，forEach 文を用いてループ処理を行う.

- 条件付きレンダリング: 本システムでは id 整合性のために空データ (null) やダミーデータ (id:0) を保持している．そのため，forEach ループ内で条件分岐を行い，有効なデータ（null かつ id:0 でないもの）のみを HTML テーブルとして出力する．

- 要素: 各行ごとの要素の名前に対応する詳細リンク，追加ボタン

4.6.3 新規作成機能

- フォーム: GET /stella/create
  - 処理: 表 4.1 の id を除くすべてのプロパティを入力フィールドとして表示する．
  - 送信先: POST /stella
  - 要素: 各プロパティに対応する入力フィールド，登録ボタン，登録後新規作成ボタン

- 作成処理: POST /stella
  - リクエストボディから値を取得．
  - 新しい id を採番し，サーバー変数（配列）に push する．
  - 新規作成したデータの内容をサーバーのターミナルに出力する．
  - 処理完了後，一覧画面 (/stella) へリダイレクトする．

4.6.4 詳細表示機能

- URL: /stella/:id

- 処理: URL パラメータの id に基づき配列を検索し，対象データを表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/stella) へリダイレクトする．

- 要素: 編集ボタン，削除ボタン，一覧に戻るリンク

4.6.5 編集・更新機能

- フォーム: GET /stella/edit/:id
  - 処理: 対象データを検索し，value 属性に現在の値を埋め込んで表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/stella) へリダイレクトする．

  - 送信先: POST /stella/update/:id

- 更新処理: POST /stella/update/:id
  - id に基づき配列内の該当要素を特定.
  - リクエストボディの値でプロパティを上書きする.
  - 表示: 更新内容をサーバーのターミナルに出力する.
  - 更新後，詳細画面 (/stella/:id) を表示する.

4.6.6 削除機能

- フォーム: GET /stella/check/:id
  - 簡易確認フォームを表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/stella) へリダイレクトする.
  - 確認後、GET /stella/delete/:id で削除処理を実行する.

- 削除処理: GET /stella/delete/:id
  - id に対応する配列要素に null を代入し，データを削除状態にする（配列の要素数は変更しない）．削除対象の指定した id に対応するデータが元から存在しない場合は，一覧表示ページ (/stella) へリダイレクトする.
  - 削除した要素名をコンソールに出力する.
  - 削除完了後，一覧画面 (/stella) へリダイレクトする.

第5章　開発者向け仕様書：元素表示システム

5.1　概要

本仕様書は，Node.js およびテンプレートエンジン EJS を用いた「元素表示システム」の設計仕様書である．本システムは，サーバーサイドで元素データを管理し，EJS を用いて動的に HTML を生成・表示する．データベースの利用は行わず，サーバープロセスのメモリ上（変数）でデータを保持・操作することを前提とする．

5.2　データ管理

5.2.1　データ構造

サーバー内の配列変数で管理するデータ構造は表 5.1 の通りに構成する．

表 5.1: 元素データ構造

プロパティ名	データ型	説明
id	Number	一意な識別子
name	String	元素名
code	Number	原子番号
symbol	String	元素記号
mass	Number	原子質量数
property	String	物質の性質
group	Number	族

id を除くすべてのデータは入力フォームで作成，編集が可能である．id はデータ配列の長さに基づいて自動採番するため，作成時に入力不要である．

5.2.2　表示における制約事項

本システムでは，データ id と配列の整合性を保つため，以下の仕様を採用する．

- id と配列: 配列と id の扱いを簡易化させるため，配列の 0 番目にはダミーデータを作成し，システム上では非表示とする．
- 削除処理: データを削除する際は，配列から要素を取り除くのではなく，該当のデータを null に置き換える．これにより，他のデータの配列と id がずれることを防ぐ．
- 一覧表示の制御: EJS テンプレートの forEach ループにおいて，要素がダミーデータまたは null でない場合のみ行を描画する条件分岐（if 文）を設ける．

また，画面のレイアウトおよび配色は，style.css に定義したスタイルを用いる．css は，3 つのシステムでレイアウトを統一するため，1 つのファイルで管理する．

5.3 ディレクトリ構成

本システムは，図 5.1 に示すディレクトリ構造に従って，使用するファイルを配置する．

```
webpro_submit/
    |- app/
        |- app_system.js              (メインロジック・データ変数保持)
        |- public/                    (静的ファイル)
            |- style.css              (CSS ファイル)
            |- element_new.html       (新規作成フォーム)
        |- views/                     (EJS テンプレート)
            |- landing.ejs            (トップページ)
            |- element/               (元素システム)
                |- element_check.ejs  (削除確認画面)
                |- element_detail.ejs (詳細表示画面)
                |- element_edit.ejs   (編集フォーム)
                |- element.ejs        (一覧表示画面)
```

図 5.1: ディレクトリ構成

5.4 HTTP メソッドとルーティング

本システムにおける各 URL と HTTP メソッド，および対応する処理を表 5.2 に定義する．

表 5.2: ルーティング一覧

機能	メソッド	パス (URL)	対応ビュー
トップページ	GET	/	views/landing.ejs
一覧表示	GET	/element	views/element/element.ejs
新規作成フォーム	GET	/element/create	public/element_new.html
詳細表示	GET	/element/:id	views/element_detail.ejs
編集フォーム	GET	/element/edit/:id	views/element_edit.ejs
削除確認	GET	/element/check/:id	views/element_check.ejs
新規データ作成	POST	/element	処理後一覧を表示
新規データ作成	POST	/element/create	処理後新規作成へリダイレクト
データ更新	POST	/element/update/:id	処理後詳細ページを表示
データ削除	GET	/element/delete/:id	処理後一覧へリダイレクト

5.5 ページ遷移

本システムにおける画面間の遷移を図 5.2 に示す．なお，本システムのページには戻るリンクを配置するため，一覧表示ページ及び詳細表示ページに直接遷移することが可能である．

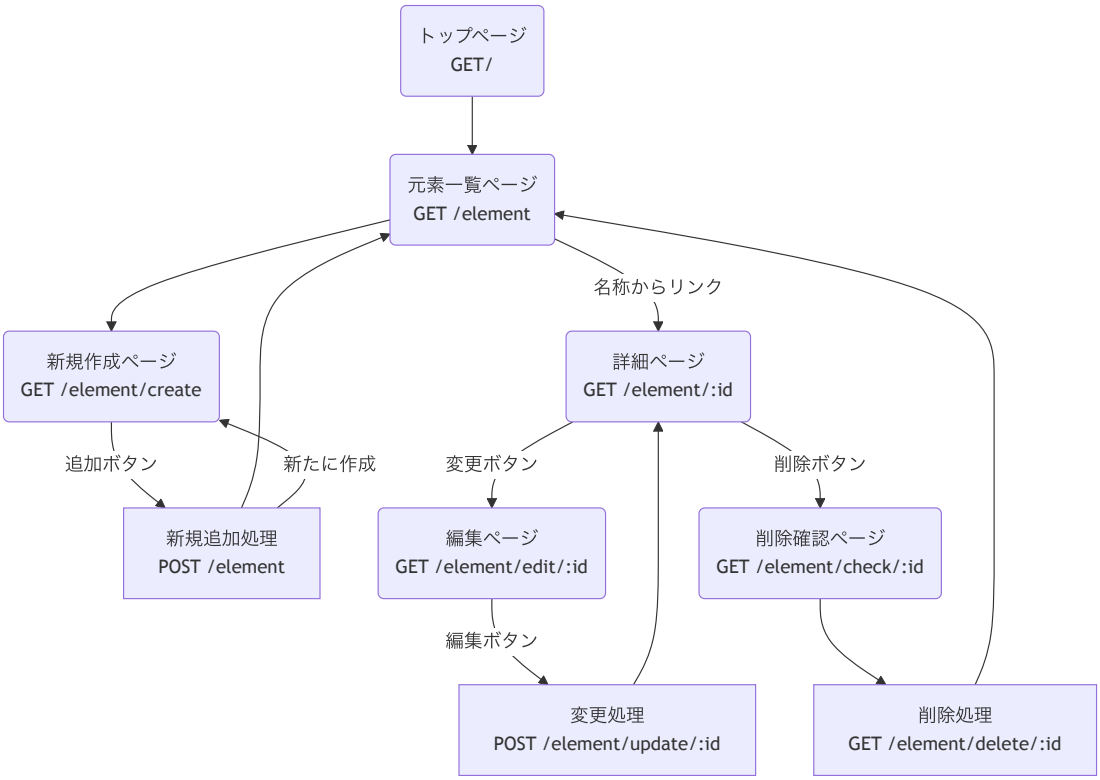

図 5.2: 画面遷移フローチャート

5.6 各機能・リソース詳細

5.6.1 トップページ

- URL: /
- 処理: views/landing.ejs を表示する．
- 要素: 各システムへのリンク

5.6.2 一覧表示機能

- URL: /element
- 処理: サーバー変数の全データを EJS に渡し，forEach 文を用いてループ処理を行う．
  - 条件付きレンダリング: 本システムでは id 整合性のために空データ (null) やダミーデータ (id:0) を保持している．そのため，forEach ループ内で条

件分岐を行い，有効なデータ（null かつ id:0 でないもの）のみを HTML テーブルとして出力する．

- 要素: 各行ごとの要素の名前に対応する詳細リンク，追加ボタン

5.6.3 新規作成機能

- フォーム: GET /element/create
  - 処理: 表 5.1 の id を除くすべてのプロパティを入力フィールドとして表示する．
  - 送信先: POST /element
  - 要素: 各プロパティに対応する入力フィールド，登録ボタン，登録後新規作成ボタン
- 作成処理: POST /element
  - リクエストボディから値を取得．
  - 新しい id を採番し，サーバー変数（配列）に push する．
  - 新規作成したデータの内容をサーバーのターミナルに出力する．
  - 処理完了後，一覧画面 (/element) へリダイレクトする．

5.6.4 詳細表示機能

- URL: /element/:id
- 処理: URL パラメータの id に基づき配列を検索し，対象データを表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/element) へリダイレクトする．
- 要素: 編集ボタン，削除ボタン，一覧に戻るリンク

5.6.5 編集・更新機能

- フォーム: GET /element/edit/:id
  - 処理: 対象データを検索し，value 属性に現在の値を埋め込んで表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/element) へリダイレクトする．
  - 送信先: POST /element/update/:id
- 更新処理: POST /element/update/:id

- id に基づき配列内の該当要素を特定.
- リクエストボディの値でプロパティを上書きする.
- 表示: 更新内容をサーバーのターミナルに出力する.
- 更新後，詳細画面 (/element/:id) を表示する.

5.6.6 削除機能

- フォーム: GET /element/check/:id
  - 簡易確認フォームを表示する．指定した id に対応するデータが存在しない場合は，一覧表示ページ (/element) へリダイレクトする.
  - 確認後、GET /element/delete/:id で削除処理を実行する.
- 削除処理: GET /element/delete/:id
  - id に対応する配列要素に null を代入し，データを削除状態にする（配列の要素数は変更しない）．削除対象の指定した id に対応するデータが元から存在しない場合は，一覧表示ページ (/element) へリダイレクトする.
  - 削除した要素名をコンソールに出力する.
  - 削除完了後，一覧画面 (/element) へリダイレクトする.