基本的なデータ表示の Web アプリ開発

2025 年 12 月 22 日

1　はじめに

本報告書では，一覧表示を中心とした web アプリケーションである黄金裔システム，原神星 5 聖遺物システム，鳴潮限定星 5 所持キャラシステムの 3 種類開発した．各アプリケーションは，データを一覧形式で提示することで利用者が情報を直感的に把握できる構成とし，基本的な操作性と拡張性を考慮して設計している．本レポートでは、作成した Web アプリケーションについて，利用者，管理者，開発者それぞれの視点から機能の説明を行う．

2　利用者向け

以下に利用者向けの説明を行う．

2.1　概要

利用者の視点から見た本 Web アプリケーションは，登録された情報を一覧形式で閲覧・確認することを目的としたシステムである．利用者は，一覧表示されたデータを通して全体の状況を把握でき，必要に応じて個々の情報にアクセスすることが可能である．直感的な操作性を重視し，専門的な知識を必要とせずに利用できる点が特徴である．

2.2　使用できる機能

使用できる機能として，一覧表示，詳細表示，データ追加，データ削除，データ編集がある．これらの操作は統一しているため，３つのシステムにの操作に大きな差はない．

2.3　システムの使い方について

例として，黄金裔システムの使用方法を以下に示す．

2.3.1　起動画面と一覧表示

起動と同時に以下のような一覧表示が表示される．この画面では黄金裔の ID，因子個体識別コード，キャラ名が確認できる．

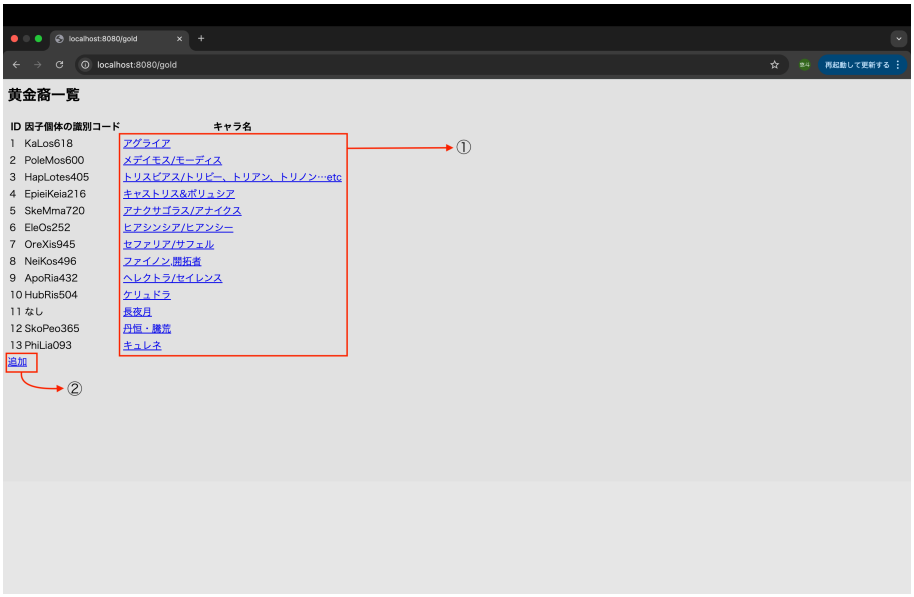

図 1　一覧表示

また，このページで可能な操作は以下の 2 つである．

1. キャラ名：選択することによる詳細表示への移動
2. 追加リンク：選択することによるデータ追加画面への移動

なお，原神星 5 聖遺物システムでは図 2 のように ID，聖遺物名，2 セット効果が一覧として表示され，鳴潮所持星 5 キャラシステムでは図 3 のように ID とキャラ名が一覧として表示される．

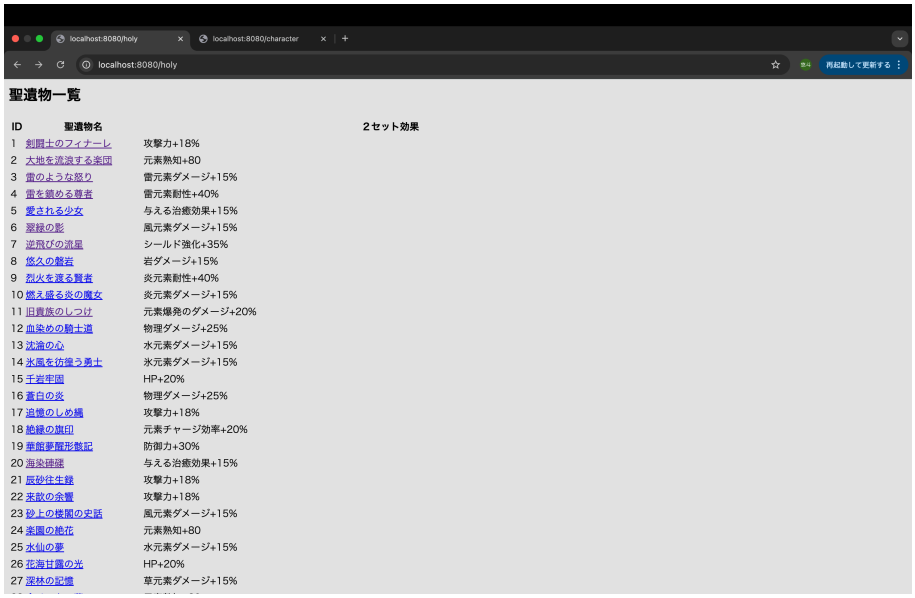

図 2　原神星 5 聖遺物システムの一覧表示

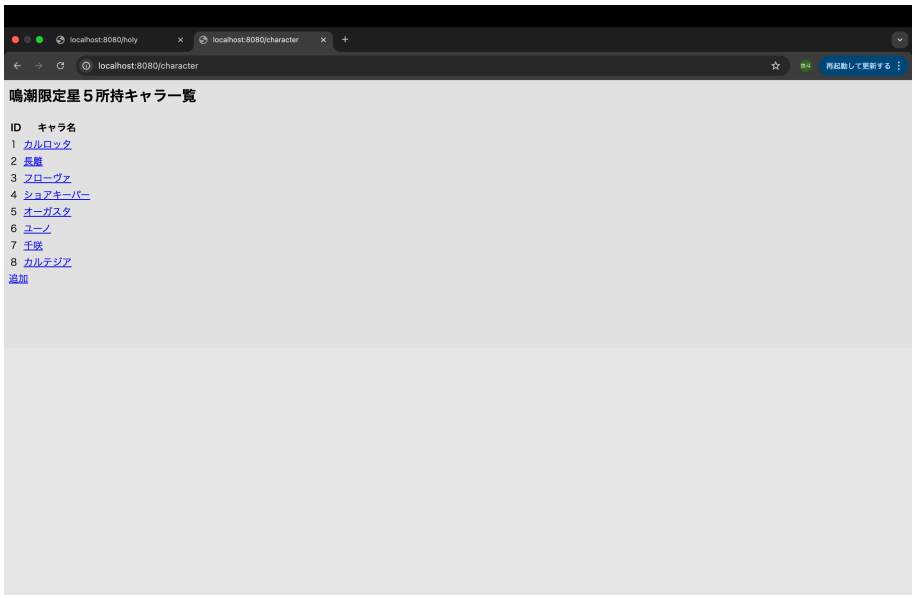

図３　鳴潮所持星５キャラシステムの一覧表示

2.3.2　詳細表示

一覧表示で任意のキャラ名を選択した後，以下のような詳細表示画面に移動する．

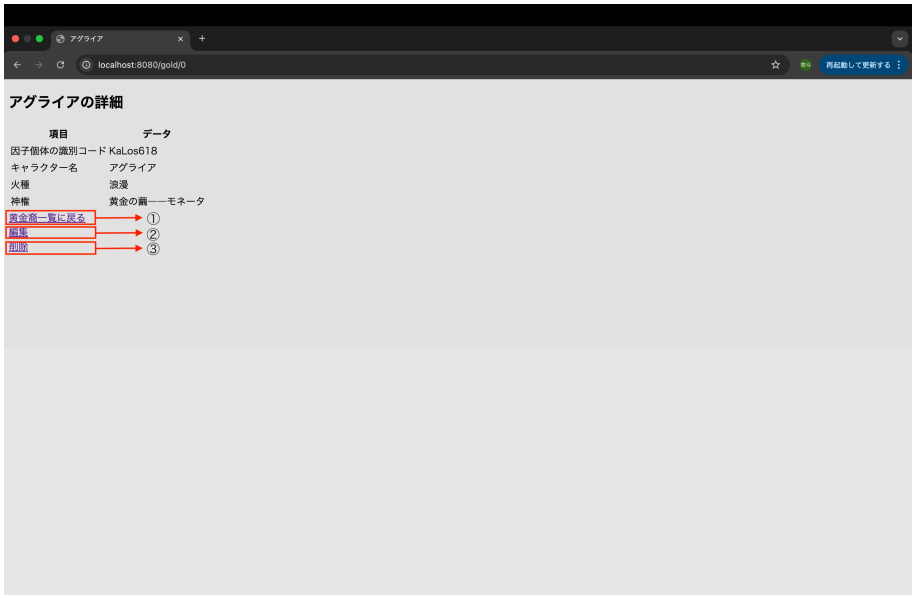

図４　詳細表示

このページでは，因子個体識別コード，キャラ名，神権が確認でき，可能な操作は以下の３つである．

1. 黄金裔一覧に戻る：一覧ページへ戻る
2. 編集：編集画面への移動

3. 削除：削除画面への移動

なお，原神星５聖遺物システムでは，図 5 のように聖遺物名，２セット効果，５セット効果，おすすめキャラクターが表示され，鳴潮限定星 5 所持キャラについてのシステムでは，図 6 のようにキャラ名，武器名，凸数，音骸セット，HP，攻撃力，防御力，クリティカル（%），クリダメ（%），共鳴効率（%），ダメバフ（有効なバフ）（%），通常攻撃 lv，スキル lv，共鳴回路 lv，共鳴解放 lv，変奏スキル lv を表示する．

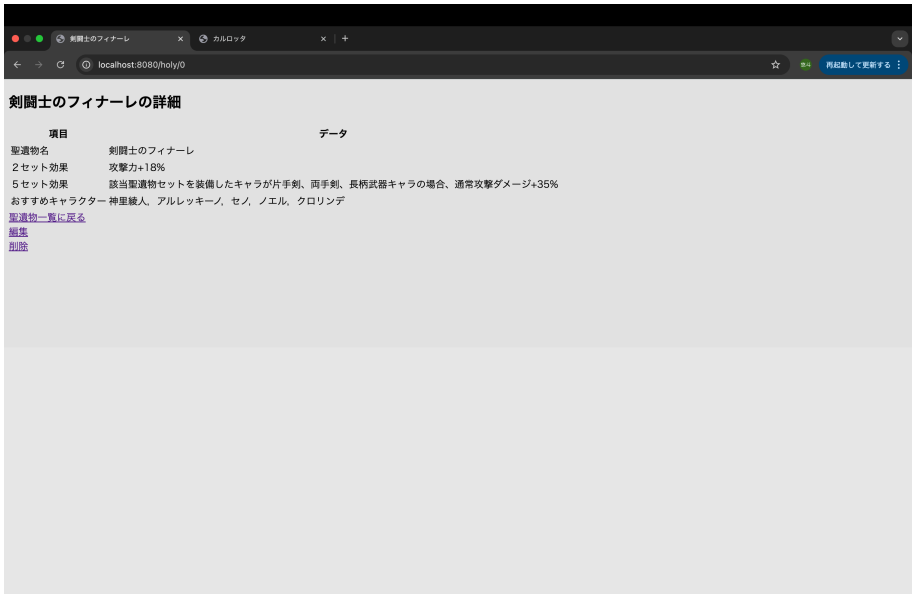

図 5　原神星５聖遺物システムの詳細表示

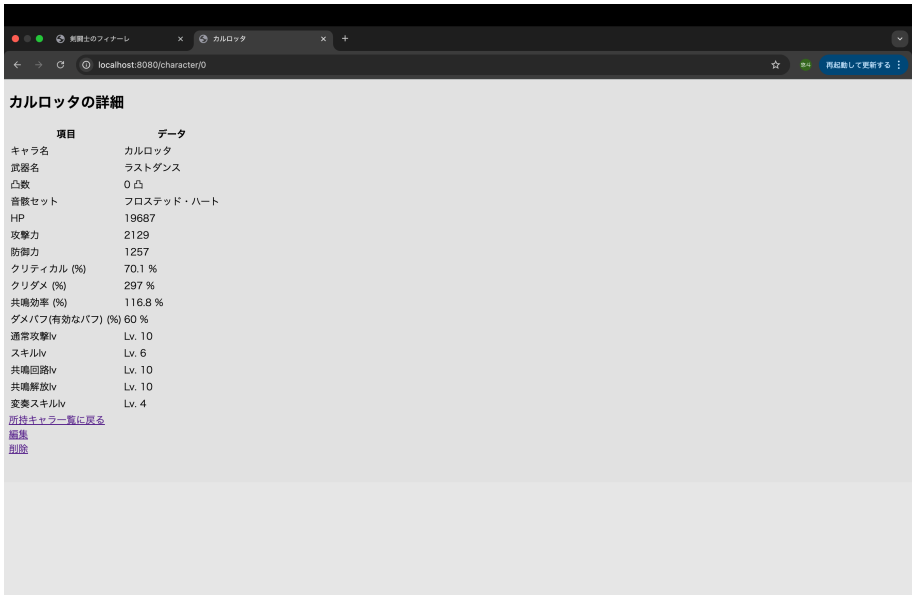

図 6　鳴潮限定星５所持キャラシステムの詳細表示

2.3.3　データ追加

一覧表示で追加リンクを選択することで，以下のようなデータ追加画面に移動する．

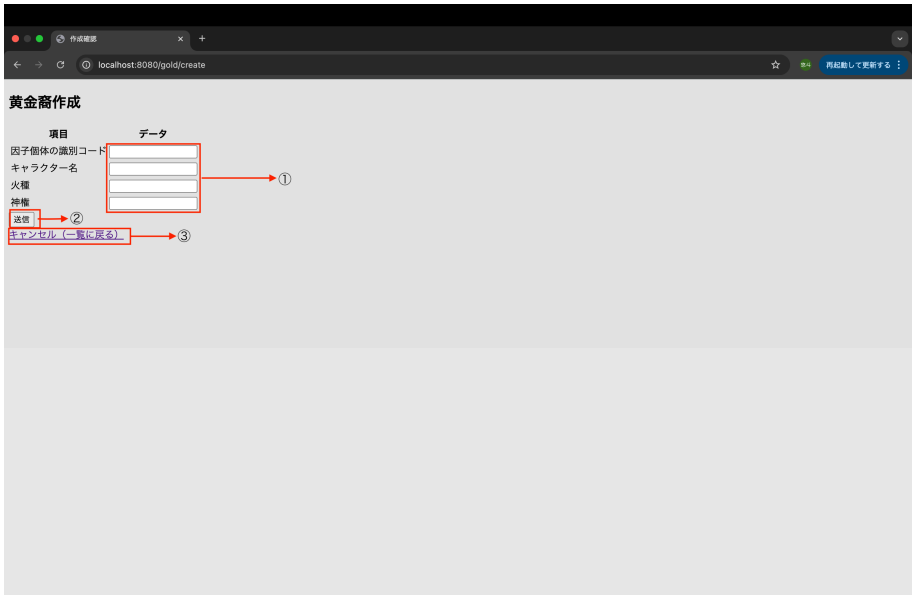

図 7　データ追加画面

1. 入力欄：追加したいキャラの詳細を入力
2. 送信ボタン：押すことで内容を決定して追加する
3. キャンセル（一覧に戻る）：一覧ページへ戻る

①に追加したいキャラの詳細を入力し，②の送信ボタンを押すことで追加することができる．また，③の一覧に戻るを押すことで一覧へ戻ることもできる．追加を決定すると一覧画面に移動し，図 8 のように，最も後ろの ID の位置に追加される．なお，他の 2 つのシステムの場合でも，入力内容が変化するだけで操作に違いはない．

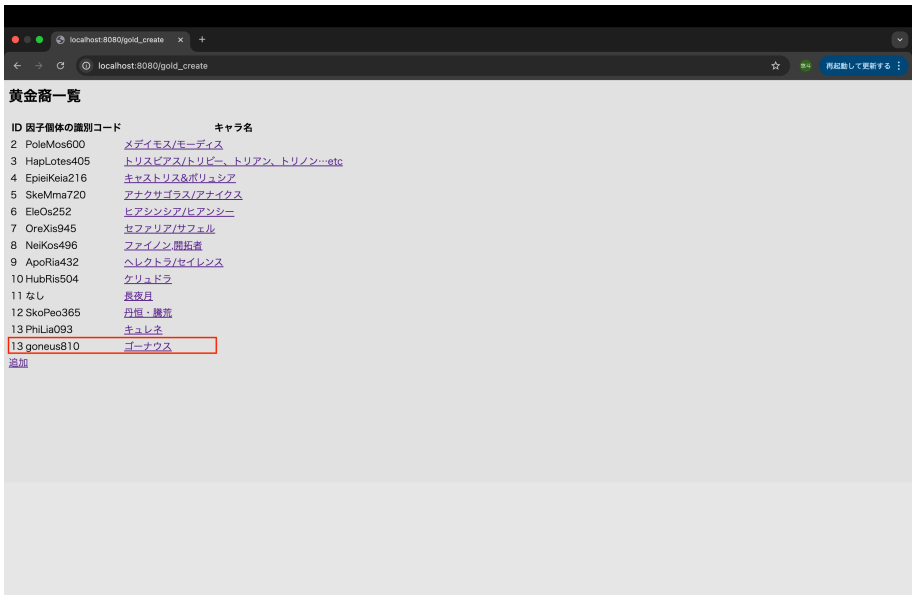

図 8　データ追加後の一覧画面

2.3.4　データ削除

詳細表示で削除リンクを押すことで，以下のようなデータ削除を行える削除画面に移動することができる．

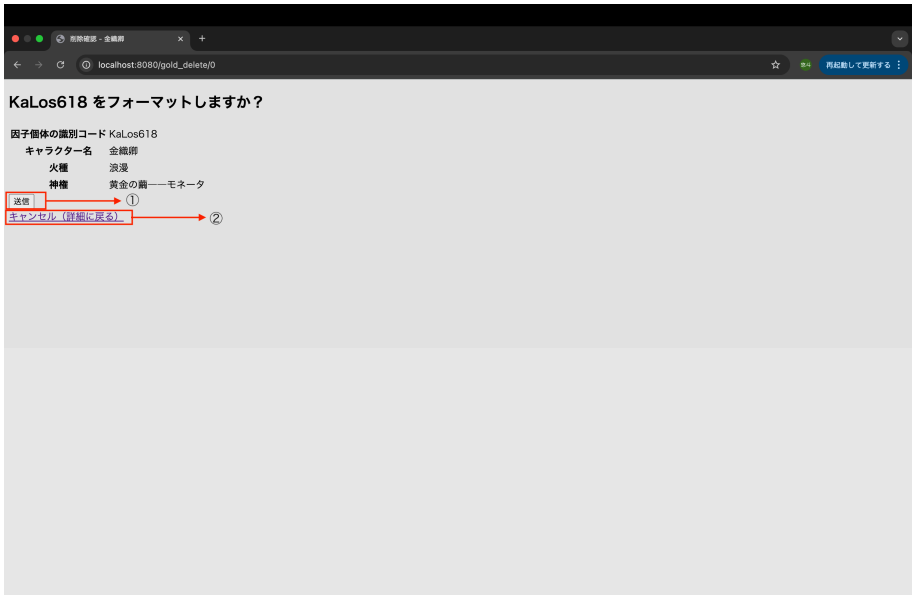

図 9　データ削除画面

1. 送信ボタン：押すことで内容を決定して追加する
2. キャンセル（詳細に戻る）：一覧ページへ戻る

削除する情報を確認し，①の送信ボタンを押すことで，削除することができる．また，②のキャンセルを押

すことで詳細表示に戻ることもできる．なお，他の２つのシステムの場合でも，表示される詳細内容が変わるだけで操作に違いはない．

2.3.5　データ編集

詳細画面で編集を押すことで，以下のようなデータの編集を行える編集画面に移動し，データの編集ができる．

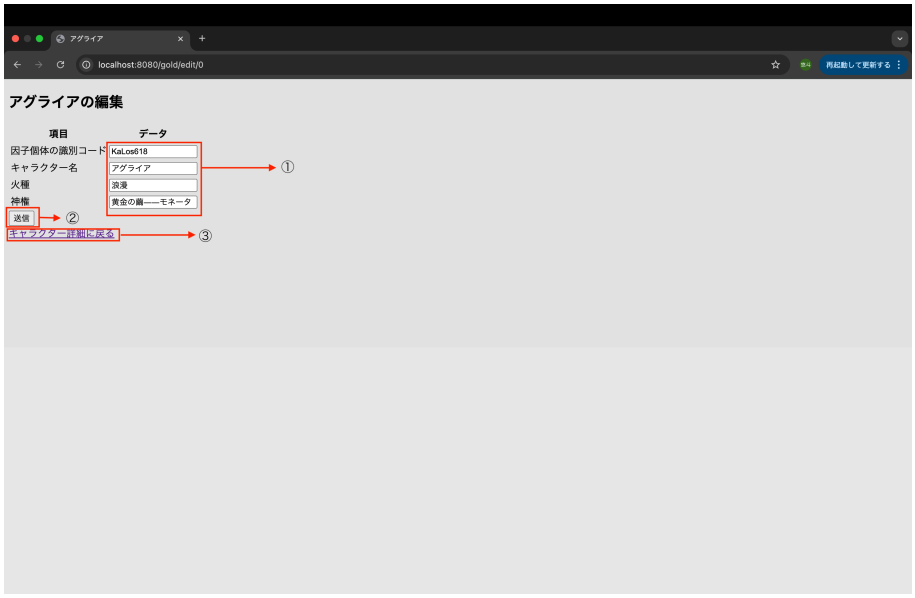

図 10　データ編集画面

1. 入力欄：編集したいキャラの詳細を入力
2. 送信ボタン：押すことで内容を決定して編集を決定する
3. キャラクター詳細に戻る：一覧ページへ戻る

編集画面では，①に変種したい部分の内容を書き換え，②の送信ボタンを押すことで，編集をすることができる．また，③のキャラクター詳細に戻るを押すことで詳細表示に戻ることもできる．例として，図 11 のようにキャラクター名を変更すると図 12 のように詳細が変更できたことを確認できる．なお，他の２つのシステムの場合でも，編集内容が変わるだけで，操作に違いはない．

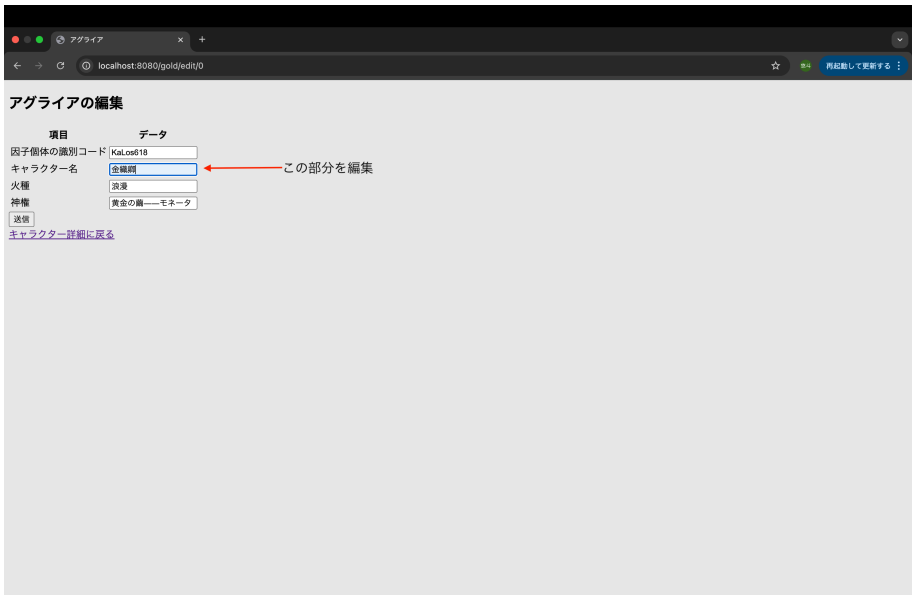

図 11　データ編集中の編集画面

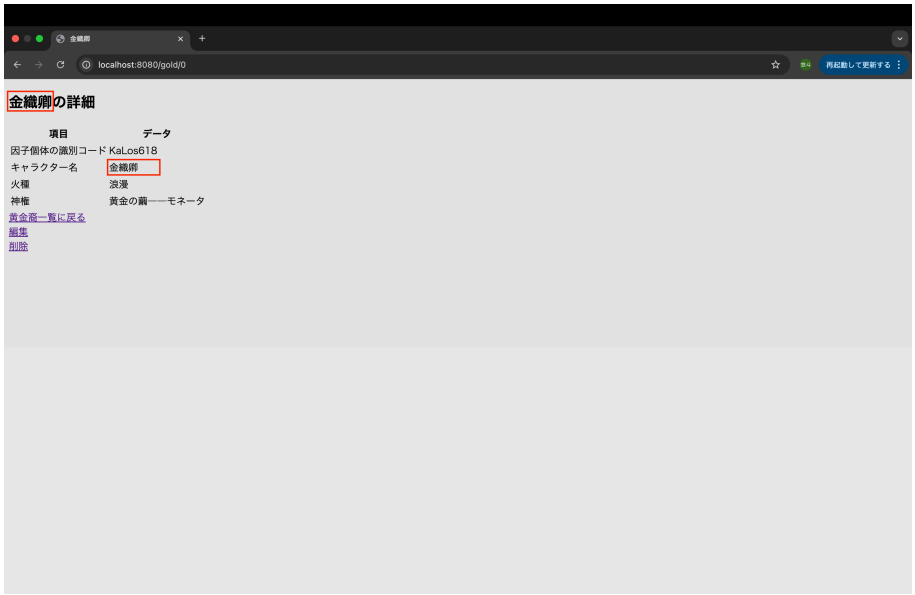

図 12　データ編集後の一覧画面

3　管理者向け

以下に管理者向けの説明を行う

3.1　サーバーの起動方法

サーバー (app5.js) の起動方法について説明する．

初めに，ターミナルで JavaScript ファイルがあるディレクトリに移動し，そこでプログラム 1 を実行する．

プログラム 1　サーバーの起動する際のプログラム

```
1    $ node app5.js
```

実行した際に Example app listening on port 8080! のように表示されればサーバーの起動が成功している．

3.2　サーバーへの接続方法

3.2.1　telnet を使用して接続する方法

telnet を使用して接続する方法について説明する．初めに，サーバーにアクセスするのにコマンド**??**のプログラムを実行する

プログラム 2　サーバーにアクセスする際のプログラム

```
1    $ telnet localhost 8080
```

次に，プログラム 3 を実行しサーバーへのリクエストを行う．なお，1 行目を書き終えた際に一回，2 行目を書き終えた際に 2 回 Enter キーを押す必要がある．

プログラム 3　サーバーにリクエストをする際のプログラム

```
1    GET /gold HTTP/1.1
2    HOST: localhost
```

3.2.2　google などでの接続

google などを使用した接続方法について説明する．URL で以下のように実行することで一覧ページにアクセスすることができる．

プログラム 4　google などでサーバーにリクエストをする際のプログラム

```
1    http://localhost:8080/holy
```

3.3　起動できない場合

サーバーを起動しないまま，クライアント側から接続をしようとすると，接続できない状況になることがある．

3.4　終了方法

終了する際は，サーバーを動かしているターミナルで Control + c と入力することでサーバーを終了することができる．

4　開発者向け

4.1　概要

開発者の視点から見た本 Web アプリケーションは，一覧表示を中心とした Web システムの設計および実装方法を検証するための開発対象である．ページ構造，ページ遷移，HTTP メソッドとリソース設計を明確に分離することで，保守性と拡張性を考慮した構成としている．今後の機能追加や仕様変更を容易に行える点を設計上の特徴とする．作成した 3 種類のシステムについて説明を行う．

4.2　黄金裔システムについて

4.2.1　黄金裔システムのデータ構造について

黄金裔システムのデータ構造を表 1 に示す．この配列（ougonei）の形はプログラム 5 のようになっている．

表 1　黄金裔一覧システムのデータ構造

項目名	型	内容
id	数値	キャラ ID
code	文字列	キャラクターデータ名
name	文字列	キャラクター名
spark	文字列	背負った火種
divine	文字列	担う神権

プログラム 5　配列の形

```
1     { id:1, code:"KaLos618", nameアグライア:"", spark浪漫:"", divine黄金の繭――モネータ
          :""},
```

4.2.2　黄金裔システムの遷移図

黄金裔システムのページ遷移図を表 13 に示す．

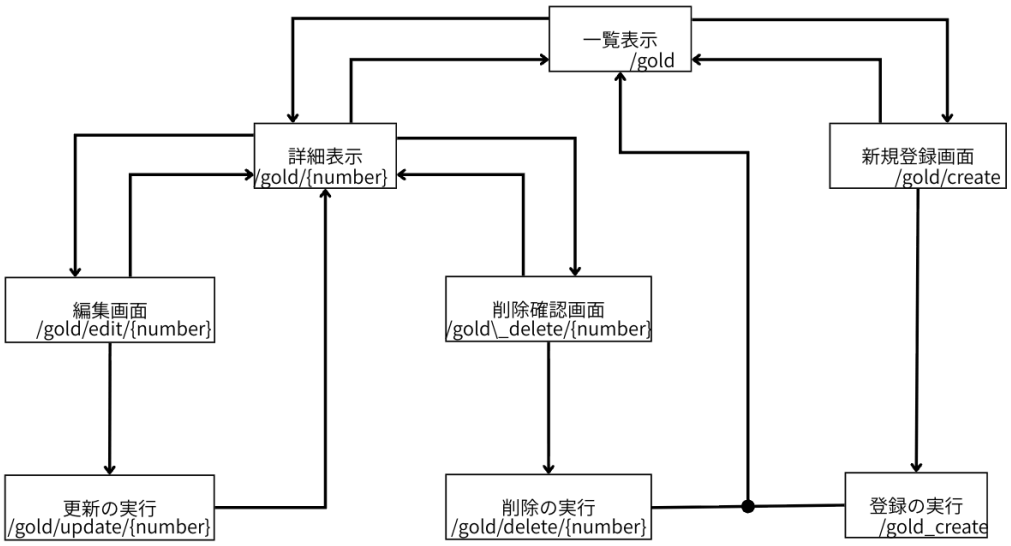

図 13　黄金裔システムの遷移図

本システムでは，一覧表示ページを起点として，各操作（詳細表示・新規登録・編集・削除）をページ内のリンクやボタンをクリックすることで遷移する構成となっている．以下に，主なページ遷移の流れを説明する．まず，利用者が 一覧表示ページ（/gold） にアクセスすると，登録されているデータのうちの，ID，因子個体の識別コード，キャラ名が表示される．この一覧ページには，キャラ名がリンクとして配置されており，任意の項目をクリックすることで，そのデータの詳細表示ページ（/gold/number）へ遷移する．詳細表示ページでは，選択したデータの内容（因子個体の識別コード，キャラ名，火種，神権）が表示されるとともに，「編集」，「削除」および「一覧に戻る」といった操作用のリンクが設置されている．編集リンクをクリックした場合は，編集画面（/gold/edit/number） に遷移し，削除リンクをクリックした場合は，削除確認画面（/gold_delete/number） に遷移する．一覧に戻るリンクをクリックした場合は，一覧ページ（/gold）に遷移する．編集画面では，既存データの内容が入力フォームに反映された状態で表示される．利用者が内容を変更し，「更新」ボタンを押下すると，更新の実行（/gold/update/number）が行われ，処理完了後は詳細表示ページへ戻る．なお，一覧に戻るリンクも存在する．削除確認画面では，削除対象のデータ内容が再度表示され，利用者が削除を確定すると，削除の実行（/gold/delete/number） が行われる．削除せずに戻る場合は，キャンセル（一覧に戻る）リンク（/gold）を押下することで一覧表示ページに遷移する．削除処理完了後は，自動的に一覧表示ページへ遷移し，最新のデータ状態が表示される．また，一覧表示ページには「新規登録」へのリンクが設置されており，これをクリックすることで 新規登録画面（/gold/create） に遷移する．新規登録画面で必要事項を入力し，登録ボタンを押下すると，登録の実行（/gold_create） が行われ，登録完了後は一覧表示ページへ戻る．ここでも新規登録をしない場合は一覧に戻るリンクを押下することで一覧表示ページに遷移する．このように，本システムでは直感的なページ遷移を実現している．

4.3　黄金裔システムの HTTP メソッドとリソース名一覧

本説では，HTTP メソッドとリソース名一覧について説明していく．黄金裔システムの HTTP メソッドとリソース名一覧を表 14 に示す．

図 14　HTTP メソッドとリソース名一覧（黄金裔システム）

HTTP メソッド	リソース名	内容
GET	/gold	一覧の取得
GET	/gold/number	指定した詳細を取得
GET	/gold/create	新規作成画面を表示
POST	/gold_create	新規登録の実行
GET	/gold/edit/number	編集画面の表示
POST	/gold/update/number	更新の実行
GET	/gold_delete/number	削除確認画面の表示
GET	/gold/delete/number	削除の実行

4.4　黄金裔システムのリソース名ごとの機能の詳細

4.4.1　一覧の取得（/gold）について

一覧の取得（/gold）の機能の詳細についてプログラム 6 を参照しながら説明を行う．

プログラム 6　一覧の取得（/gold）

```
1    app.get("/gold", (req, res) => {
2        res.render('gold', {data: ougonei} );
3    });
```

プログラム 6 の 1 行目から，メソッドが GET であることがわかる．2 行目から，ougonei 配列（データ構造）を取得し，ejs テンプレート gold.ejs にデータを渡す．これにより，一覧ページが出力される．

4.4.2　指定した詳細を取得（/gold/number）について

指定した詳細を取得（/gold/number）の機能の詳細についてプログラム 7 を参照しながら説明を行う．

プログラム 7　指定した詳細を取得（/gold/number）

```
1    app.get("/gold/:number", (req, res) => {
2        const number = req.params.number;
3        const detail = ougonei[ number ];
4        res.render('gold_detail', {id: number, data: detail} );
5    });
```

プログラム 7 の 1 行目から，メソッドが GET，パスパラメーターの":number" があることがわかる．2 行目で URL から指定された番号を取得し，3 行目で受け取った番号を元に配列 ougonei から対応する要素を取

得する．４行目で取得したデータを詳細表示用テンプレートに渡すことで，対象となるデータの詳細ページを出力する．

4.4.3　新規作成画面を表示（/gold/create）について

新規作成画面を表示（/gold/create）の機能の詳細についてプログラム 8 を参照しながら説明を行う．

プログラム 8　新規作成画面を表示（/gold/create）

```
1    app.get("/gold/create", (req, res) => {
2        res.render('gold_create');
3    });
```

プログラム 8 では，１行目で GET メソッドによる/gold/create へのアクセスを入力として受け取り，サーバ側では処理を行わず，２行目で入力フォームを含むページをそのまま出力する．

4.4.4　新規登録の実行（/gold_create）について

新規登録の実行（/gold_create）の機能の詳細についてプログラム 9 を参照しながら説明を行う．

プログラム 9　新規登録の実行（/gold_create）

```
1    app.post("/gold_create", (req, res) => {
2        const id = ougonei.length + 1;
3        const code = req.body.code;
4        const name = req.body.name;
5        const spark = req.body.spark;
6        const divine = req.body.divine;
7        ougonei.push( { id: id, code: code, name: name, spark: spark, divine: divine }
             );
8        console.log( ougonei );
9        res.render('gold', {data: ougonei} );
10   });
```

プログラム 9 では１行目で POST メソッドによって送信されたフォーム入力を受け取り，２行目で，新規 ID の作成，3〜６行目で入力された内容を受け取り，７行目で配列に格納することで，新規データとして配列 ougonei に追加する処理を行う．なお，８行目は動作確認用のデバック出力である．データ追加後は，更新されたデータ一覧を一覧表示用ページとして出力し，利用者が登録結果を即座に確認できるようにしている．

4.4.5　編集画面の表示（/gold/edit/number）について

編集画面の表示（/gold/edit/number）の機能の詳細についてプログラム 10 を参照しながら説明を行う．

プログラム 10　編集画面の表示（/gold/edit/number）

```
1    app.get("/gold/edit/:number", (req, res) => {
2        const number = req.params.number;
3        const detail = ougonei[ number ];
4        res.render('gold_edit', {id: number, data: detail} );
5    });
```

プログラム 10 では，既存データを編集するための入力画面を表示することを目的としている．1 行目で，GET メソッドによる/gold/edit/:number へのアクセスを入力として受け取り，2，3 行目で指定された番号に対応するデータを取得した上で，編集用フォームを含むページを出力する．

4.4.6 更新の実行（/gold/update/number）について

更新の実行（/gold/update/number）の機能の詳細についてプログラム 11 を参照しながら説明を行う．

プログラム 11 更新の実行（/gold/update/number）

```
1    app.post("/gold/update/:number", (req, res) => {
2        const number = req.params.number;
3        ougonei[req.params.number].code = req.body.code;
4        ougonei[req.params.number].name = req.body.name;
5        ougonei[req.params.number].spark = req.body.spark;
6        ougonei[req.params.number].divine = req.body.divine;
7        console.log( ougonei );
8        res.redirect(‘/gold/${number}‘);
9    });
```

プログラム 11 では，POST メソッドによって送信された編集内容を受け取り，パスパラメータで指定されたデータを更新する処理を行う．2 行目で，更新する対象のデータ番号を受け取り，3〜6 行目で入力された更新内容を受け取り，配列に新しく格納する．更新完了後は 8 行目で詳細表示へリダイレクトすることで，利用者が変更内容を即座に確認できるようにしている．なお，7 行目は，動作確認用のデバック出力である．

4.4.7 削除確認画面の表示（/gold_delete/number）について

削除確認画面の表示（/gold_delete/number）の機能の詳細についてプログラム 12 を参照しながら説明を行う．

プログラム 12 削除確認画面の表示（/gold_delete/number）

```
1    app.get("/gold_delete/:number", (req, res) => {
2        const number = req.params.number;
3        const detail = ougonei[ number ];
4        res.render(’gold_delete’, {id: number, data: detail} );
5    });
```

プログラム 9 では，データ削除を実行する前に内容を確認するための画面表示する機能をを持つ．1 行目では，GET メソッドによる/gold_delete/:number へのアクセスを入力として受け取り，2，3 行目で指定された番号に対応するデータを取得した上で，4 行目で，削除確認用ページを出力する．

4.4.8 削除の実行（/gold/delete/number）について

削除の実行（/gold/delete/number）の機能の詳細についてプログラム 13 を参照しながら説明を行う．

プログラム 13 削除の実行（/gold/delete/number）

```
1    app.get("/gold/delete/:number", (req, res) => {
2        ougonei.splice( req.params.number, 1 );
```

```
3         res.redirect('/gold' );
4     });
```

プログラム 13 では，パスパラメータで指定されたデータを配列から削除する処理を行う．２行目で，受け取った番号の配列の内容を削除する．削除完了後は一覧表示ページへリダイレクトすることで，最新のデータ状態を利用者に提示する．

4.5 原神星５聖遺物システム

4.5.1 原神星５聖遺物システムのデータ構造について

原神星 5 聖遺物システムのデータ構造を表 2 に示す．この配列（relic）の形はプログラム 14 のようになっている．

表 2 原神星５聖遺物システムのデータ構造

項目名	型	内容
id	数値	聖遺物 ID
name	文字列	聖遺物名
two_set	文字列	2 セット効果
five_set	文字列	5 セット効果
chara	文字列	おすすめキャラ

プログラム 14 配列の形

```
1     { id:1, name剣闘士のフィナーレ:"", two_set攻撃力:"+18%", five_set該当聖遺物セットを
          装備したキャラが片手剣、両手剣、長柄武器キャラの場合、通常攻撃ダメージ
          :"+35%", chara神里綾人, アルレッキーノ, セノ, ノエル, クロリンデ:""},
```

4.5.2 原神星５聖遺物システムの遷移図

原神星 5 聖遺物システムのページ遷移図を表 15 に示す．

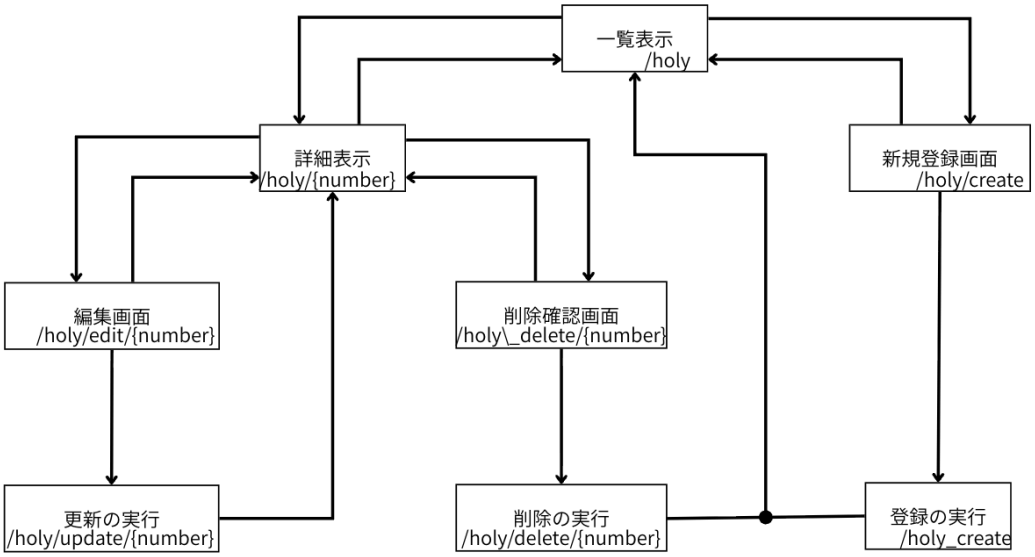

図 15　鳴潮限定星 5 所持キャラシステムの遷移図

本システムでは，一覧表示ページを起点として，各操作（詳細表示・新規登録・編集・削除）をページ内のリンクやボタンをクリックすることで遷移する構成となっている．以下に，主なページ遷移の流れを説明する．まず，利用者が一覧表示ページ（/holy）にアクセスすると，登録されているデータのうちの，ID，聖遺物名，２セット効果が表示される．この一覧ページには，聖遺物名がリンクとして配置されており，任意の項目をクリックすることで，そのデータの詳細表示ページ（/holy/number）へ遷移する．詳細表示ページでは，選択したデータの内容（聖遺物名，２セット効果，５セット効果）が表示されるとともに，「編集」，「削除」および「一覧に戻る」といった操作用のリンクが設置されている．編集リンクをクリックした場合は，編集画面（/holy/edit/number）に遷移し，削除リンクをクリックした場合は，削除確認画面（/holy_delete/number）に遷移する．一覧に戻るリンクをクリックした場合は，一覧ページ（/holy）に遷移する．編集画面では，既存データの内容が入力フォームに反映された状態で表示される．利用者が内容を変更し，「更新」ボタンを押下すると，更新の実行（/holy/update/number）が行われ，処理完了後は詳細表示ページへ戻る．なお，一覧に戻るリンクも存在する．削除確認画面では，削除対象のデータ内容が再度表示され，利用者が削除を確定すると，削除の実行（/holy/delete/number）が行われる．削除せずに戻る場合は，キャンセル（一覧に戻る）リンク（/holy）を押下することで一覧表示ページに遷移する．削除処理完了後は，自動的に一覧表示ページへ遷移し，最新のデータ状態が表示される．また，一覧表示ページには「新規登録」へのリンクが設置されており，これをクリックすることで新規登録画面（/holy/create）に遷移する．新規登録画面で必要事項を入力し，登録ボタンを押下すると，登録の実行（/holy_create）が行われ，登録完了後は一覧表示ページへ戻る．ここでも新規登録をしない場合は一覧に戻るリンクを押下することで一覧表示ページに遷移する．このように，本システムでは直感的なページ遷移を実現している．

4.6　黄金裔システムの HTTP メソッドとリソース名一覧

本説では，HTTP メソッドとリソース名一覧について説明していく．黄金裔システムの HTTP メソッドとリソース名一覧を表 16 に示す．

図 16　HTTP メソッドとリソース名一覧（黄金裔システム）

HTTP メソッド	リソース名	内容
GET	/holy	一覧の取得
GET	/holy/number	指定した詳細を取得
GET	/holy/create	新規作成画面を表示
POST	/holy_create	新規登録の実行
GET	/holy/edit/number	編集画面の表示
POST	/holy/update/number	更新の実行
GET	/holy_delete/number	削除確認画面の表示
GET	/holy/delete/number	削除の実行

4.7　原神星５聖遺物システムのリソース名ごとの機能の詳細

以下にリソース名ごとの機能の詳細を示す．

4.7.1　一覧の取得（/holy）について

一覧の取得（/holy）の機能の詳細についてプログラム 15 を参照しながら説明を行う．

プログラム 15　一覧の取得（/holy）

```
1 app.get("/holy", (req, res) => {
2     res.render('holy', {data: relic} );
3 });
```

プログラム 15 の 1 行目から，メソッドが GET であることがわかる．２行目から，relic 配列（データ構造）を取得し，ejs テンプレート holy.ejs にデータを渡す．これにより，一覧ページが出力される．

4.7.2　指定した詳細を取得（/holy/number）について

指定した詳細を取得（/holy/number）の機能の詳細についてプログラム 16 を参照しながら説明を行う．

プログラム 16　指定した詳細を取得（/holy/number）

```
1 app.get("/holy/:number", (req, res) => {
2     const number = req.params.number;
3     const detail = relic[ number ];
4     res.render('holy_detail', {id: number, data: detail} );
5 });
```

プログラム 16 の 1 行目から，メソッドが GET，パスパラメーターの":number" があることがわかる．2 行目で URL から指定された番号を取得し，3 行目で受け取った番号を元に配列 relic から対応する要素を取得する．4 行目で取得したデータを詳細表示用テンプレートに渡すことで，対象となるデータの詳細ページを出力する．

4.7.3 新規作成画面を表示（/holy/create）について

新規作成画面を表示（/holy/create）の機能の詳細についてプログラム 17 を参照しながら説明を行う．

プログラム 17 新規作成画面を表示（/holy/create）

```
1 app.get("/holy/create", (req, res) => {
2     res.render('holy_create');
3 });
```

プログラム 17 では，1 行目で GET メソッドによる/holy/create へのアクセスを入力として受け取り，サーバ側では処理を行わず，2 行目で入力フォームを含むページをそのまま出力する．

4.7.4 新規登録の実行（/holy_create）について

新規登録の実行（/holy_create）の機能の詳細についてプログラム 18 を参照しながら説明を行う．

プログラム 18 新規登録の実行（/holy_create）

```
1  app.post("/holy_create", (req, res) => {
2      const id = relic.length + 1;
3      const name = req.body.name;
4      const two_set = req.body.two_set;
5      const five_set = req.body.five_set;
6      const chara = req.body.chara;
7      relic.push( { id: id, name: name, two_set: two_set, five_set: five_set, chara:
           chara } );
8      console.log( relic );
9      res.render('holy', {data: relic} );
10 });
```

プログラム 18 では 1 行目で POST メソッドによって送信されたフォーム入力を受け取り，2 行目で，新規 ID の作成，3〜6 行目で入力された内容を受け取り，7 行目で配列に格納することで，新規データとして配列 relic に追加する処理を行う．なお，8 行目は動作確認用のデバック出力である．データ追加後は，更新されたデータ一覧を一覧表示用ページとして出力し，利用者が登録結果を即座に確認できるようにしている．

4.7.5 編集画面の表示（/holy/edit/number）について

編集画面の表示（/holy/edit/number）の機能の詳細についてプログラム 19 を参照しながら説明を行う．

プログラム 19 編集画面の表示（/holy/edit/number）

```
1 app.get("/holy/edit/:number", (req, res) => {
2     const number = req.params.number;
3     const detail = relic[ number ];
```

```
4     res.render('holy_edit', {id: number, data: detail} );
5 });
```

プログラム 19 では，既存データを編集するための入力画面を表示することを目的としている．1 行目で，GET メソッドによる/holy/edit/:number へのアクセスを入力として受け取り，2，3 行目で指定された番号に対応するデータを取得した上で，編集用フォームを含むページを出力する．

4.7.6 更新の実行（/holy/update/number）について

更新の実行（/holy/update/number）の機能の詳細についてプログラム 20 を参照しながら説明を行う．

プログラム 20　更新の実行（/holy/update/number）

```
1 app.post("/holy/update/:number", (req, res) => {
2     const number = req.params.number;
3     relic[req.params.number].name = req.body.name;
4     relic[req.params.number].two_set = req.body.two_set;
5     relic[req.params.number].five_set = req.body.five_set;
6     relic[req.params.number].chara = req.body.chara;
7     console.log( relic );
8     res.redirect(`/holy/${number}`);
9 });
```

プログラム 20 では，POST メソッドによって送信された編集内容を受け取り，パスパラメータで指定されたデータを更新する処理を行う．2 行目で，更新する対象のデータ番号を受け取り，3〜6 行目で入力された更新内容を受け取り，配列に新しく格納する．更新完了後は 8 行目で詳細表示へリダイレクトすることで，利用者が変更内容を即座に確認できるようにしている．なお，7 行目は，動作確認用のデバック出力である．

4.7.7 削除確認画面の表示（/holy_delete/number）について

削除確認画面の表示（/holy_delete/number）の機能の詳細についてプログラム 21 を参照しながら説明を行う．

プログラム 21　削除確認画面の表示（/holy_delete/number）

```
1 app.get("/holy_delete/:number", (req, res) => {
2     const number = req.params.number;
3     const detail = relic[ number ];
4     res.render('holy_delete', {id: number, data: detail} );
5 });
```

プログラム 21 では，データ削除を実行する前に内容を確認するための画面表示する機能をを持つ．1 行目では，GET メソッドによる/holy_delete/:number へのアクセスを入力として受け取り，2，3 行目で指定された番号に対応するデータを取得した上で，4 行目で，削除確認用ページを出力する．

4.7.8 削除の実行（/holy/delete/number）について

削除の実行（/holy/delete/number）の機能の詳細についてプログラム 22 を参照しながら説明を行う．

プログラム 22　削除の実行（/holy/delete/number）

```
app.get("/holy/delete/:number", (req, res) => {
    relic.splice( req.params.number, 1 );
    res.redirect('/holy' );
});
```

プログラム 22 では，パスパラメータで指定されたデータを配列から削除する処理を行う．２行目で，受け取った番号の配列の内容を削除する．削除完了後は一覧表示ページへリダイレクトすることで，最新のデータ状態を利用者に提示する．

4.8　鳴潮限定星５所持キャラシステムについて

4.8.1　鳴潮限定所持星 5 キャラシステムのデータ構造

鳴潮限定所持星 5 キャラシステムのデータ構造を表 3 に示す．この配列（states）の形はプログラム 23 のようになっている．

表 3　鳴潮限定星 5 所持キャラのデータ構造

項目名	型	内容
id	数値	キャラ ID
name	文字列	キャラ名
weapon	文字列	武器名
chain	数値	凸数
sound	文字列	装備音骸
H	数値	HP
A	数値	攻撃力
B	数値	防御力
C_H	数値	クリティカル率
C_D	数値	クリティカルダメージ
charge	数値	共鳴効率
effect	数値	属性ダメージバフ
usual	数値	通常攻撃倍率
skill	数値	スキル倍率
circuit	数値	共鳴回路倍率
release	数値	共鳴解放倍率
variation	数値	変奏倍率

プログラム 23　配列の形

```
    let states =[
        {id:1, nameカルロッタ:"", weaponラストダンス:"", chain:0, soundフロステッド・ハート:"", H:19687, A:2129, B:1257, C_H:70.1, C_D:297.0, charge:116.8,
        effect:60.0, usual:10, skill:6, circuit:10, release:10, variation:4},
```

```
3           :
4           :
5     ];
```

4.8.2　鳴潮限定星５所持キャラシステムの遷移図

鳴潮限定所持星 5 キャラシステムのページ遷移図を表 17 に示す．

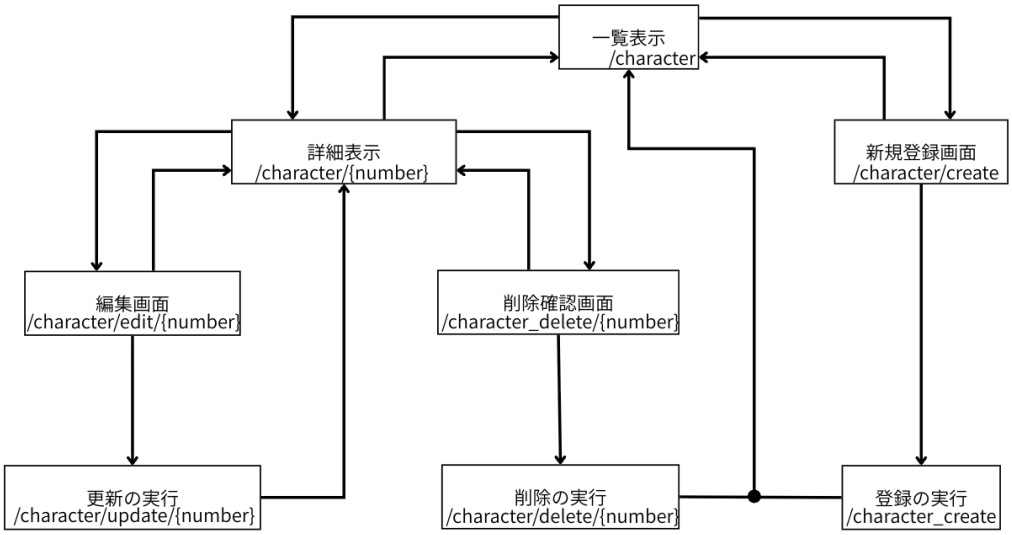

図 17　鳴潮限定星５所持キャラシステムの遷移図

本システムでは，一覧表示ページを起点として，各操作（詳細表示・新規登録・編集・削除）をページ内のリンクやボタンをクリックすることで遷移する構成となっている．以下に，主なページ遷移の流れを説明する．まず，利用者が一覧表示ページ（/character）にアクセスすると，登録されているデータのうちの，ID，キャラ名が表示される．この一覧ページには，聖遺物名がリンクとして配置されており，任意の項目をクリックすることで，そのデータの詳細表示ページ（/character/number）へ遷移する．詳細表示ページでは，選択したデータの内容（キャラ名，武器名，凸数，音骸セット，HP，攻撃力，防御力，クリティカル (%)，クリダメ (%)，共鳴効率 (%)，ダメバフ (有効なバフ) (%)，通常攻撃 lv，スキル lv，共鳴回路 lv，共鳴解放 lv，変奏スキル lv）が表示されるとともに，「編集」，「削除」および「一覧に戻る」といった操作用のリンクが設置されている．編集リンクをクリックした場合は，編集画面（/character/edit/number）に遷移し，削除リンクをクリックした場合は，削除確認画面（/character_delete/number）に遷移する．一覧に戻るリンクをクリックした場合は，一覧ページ（/character）に遷移する．編集画面では，既存データの内容が入力フォームに反映された状態で表示される．利用者が内容を変更し，「更新」ボタンを押下すると，更新の実行（/character/update/number）が行われ，処理完了後は詳細表示ページへ戻る．なお，一覧に戻るリンクも存在する．削除確認画面では，削除対象のデータ内容が再度表示され，利用者が削除を確定すると，削除の実行（/character/delete/number）

が行われる．削除せずに戻る場合は，キャンセル（一覧に戻る）リンク（/character）を押下することで一覧表示ページに遷移する．削除処理完了後は，自動的に一覧表示ページへ遷移し，最新のデータ状態が表示される．また，一覧表示ページには「新規登録」へのリンクが設置されており，これをクリックすることで 新規登録画面（/character/create）に遷移する．新規登録画面で必要事項を入力し，登録ボタンを押下すると，登録の実行（/character_create）が行われ，登録完了後は一覧表示ページへ戻る．ここでも新規登録をしない場合は一覧に戻るリンクを押下することで一覧表示ページに遷移する．このように，本システムでは直感的なページ遷移を実現している．

4.9　黄金裔システムの HTTP メソッドとリソース名一覧

本説では，HTTP メソッドとリソース名一覧について説明していく．黄金裔システムの HTTP メソッドとリソース名一覧を表 18 に示す．

図 18　HTTP メソッドとリソース名一覧（黄金裔システム）

HTTP メソッド	リソース名	内容
GET	/character	一覧の取得
GET	/character/number	指定した詳細を取得
GET	/character/create	新規作成画面を表示
POST	/character_create	新規登録の実行
GET	/character/edit/number	編集画面の表示
POST	/character/update/number	更新の実行
GET	/character_delete/number	削除確認画面の表示
GET	/character/delete/number	削除の実行

4.10　鳴潮限定星５所持キャラシステムのリソース名ごとの機能の詳細

以下にリソース名ごとの機能の詳細を示す．

4.10.1　一覧の取得（/character）について

一覧の取得（/character）の機能の詳細についてプログラム 24 を参照しながら説明を行う．

プログラム 24　一覧の取得（/character）

```
1 app.get("/character", (req, res) => {
2     res.render('character', {data: states} );
3 });
```

プログラム 24 の 1 行目から，メソッドが GET であることがわかる．２行目から，states 配列（データ構造）を取得し，ejs テンプレート character.ejs にデータを渡す．これにより，一覧ページが出力される．

4.10.2　指定した詳細を取得（/character/number）について

指定した詳細を取得（/character/number）の機能の詳細についてプログラム 25 を参照しながら説明を行う．

プログラム 25　指定した詳細を取得（/character/number）

```
1 app.get("/character/:number", (req, res) => {
2     const number = req.params.number;
3     const detail = states[ number ];
4     res.render('character_detail', {id: number, data: detail} );
5 });
```

プログラム 25 の 1 行目から，メソッドが GET，パスパラメーターの":number" があることがわかる．2 行目で URL から指定された番号を取得し，3 行目で受け取った番号を元に配列 states から対応する要素を取得する．4 行目で取得したデータを詳細表示用テンプレートに渡すことで，対象となるデータの詳細ページを出力する．

4.10.3　新規作成画面を表示（/character/create）について

新規作成画面を表示（/character/create）の機能の詳細についてプログラム 26 を参照しながら説明を行う．

プログラム 26　新規作成画面を表示（/character/create）

```
1 app.get("/character/create", (req, res) => {
2     res.render('character_create');
3 });
```

プログラム 26 では，1 行目で GET メソッドによる/character/create へのアクセスを入力として受け取り，サーバ側では処理を行わず，2 行目で入力フォームを含むページをそのまま出力する．

4.10.4　新規登録の実行（/character_create）について

新規登録の実行（/character_create）の機能の詳細についてプログラム 27 を参照しながら説明を行う．

プログラム 27　新規登録の実行（/character_create）

```
1 app.post("/character_create", (req, res) => {
2     const id = states.length + 1;
3     const name = req.body.name;
4     const weapon = req.body.weapon;
5     const chain = req.body.chain;
6     const sound = req.body.sound;
7     const H = req.body.H;
8     const A = req.body.A;
9     const B = req.body.B;
10    const C_H = req.body.C_H;
11    const C_D = req.body.C_D;
12    const charge = req.body.charge;
13    const effect = req.body.effect;
```

```
14      const usual = req.body.usual;
15      const skill = req.body.skill;
16      const circuit = req.body.circuit;
17      const release = req.body.release;
18      const variation = req.body.variation;
19      states.push({ ... });
20      console.log(states);
21      res.render('character', {data: states});
22  });
```

プログラム 18 では 1 行目で POST メソッドによって送信されたフォーム入力を受け取り，2 行目で，新規 ID の作成，3〜18 行目で入力された内容を受け取り，19 行目で配列に格納することで，新規データとして配列 relic に追加する処理を行う．なお，20 行目は動作確認用のデバック出力である．データ追加後は，21 行目で更新されたデータ一覧を一覧表示用ページとして出力し，利用者が登録結果を即座に確認できるようにしている．

4.10.5 編集画面の表示（/character/edit/number）について

編集画面の表示（/character/edit/number）の機能の詳細についてプログラム 28 を参照しながら説明を行う．

プログラム 28　編集画面の表示（/character/edit/number）

```
1  app.get("/character/edit/:number", (req, res) => {
2      const number = req.params.number;
3      const detail = states[ number ];
4      res.render('character_edit', {id: number, data: detail} );
5  });
```

プログラム 28 では，既存データを編集するための入力画面を表示することを目的としている．1 行目で，GET メソッドによる/character/edit/:number へのアクセスを入力として受け取り，2，3 行目で指定された番号に対応するデータを取得した上で，編集用フォームを含むページを出力する．

4.10.6 更新の実行（/character/update/number）について

更新の実行（/character/update/number）の機能の詳細についてプログラム 29 を参照しながら説明を行う．

プログラム 29　更新の実行（/character/update/number）

```
1  app.post("/character/update/:number", (req, res) => {
2      const number = req.params.number;
3      states[req.params.number].name = req.body.name;
4      states[req.params.number].weapon = req.body.weapon;
5      states[req.params.number].chain = req.body.chain;
6      states[req.params.number].sound = req.body.sound;
7      states[req.params.number].H = req.body.H;
8      states[req.params.number].A = req.body.A;
9      states[req.params.number].B = req.body.B;
10     states[req.params.number].C_H = req.body.C_H;
```

```
11      states[req.params.number].C_D = req.body.C_D;
12      states[req.params.number].charge = req.body.charge;
13      states[req.params.number].effect = req.body.effect;
14      states[req.params.number].usual = req.body.usual;
15      states[req.params.number].skill = req.body.skill;
16      states[req.params.number].circuit = req.body.circuit;
17      states[req.params.number].release = req.body.release;
18      states[req.params.number].variation = req.body.variation;
19      console.log(states);
20      res.redirect(`/character/${number}`);
21  });
```

プログラム 29 では，POST メソッドによって送信された編集内容を受け取り，パスパラメータで指定されたデータを更新する処理を行う．２行目で，更新する対象のデータ番号を受け取り，3〜18 行目で入力された更新内容を受け取り，配列に新しく格納する．更新完了後は 20 行目で詳細表示へリダイレクトすることで，利用者が変更内容を即座に確認できるようにしている．なお，19 行目は，動作確認用のデバック出力である．

4.10.7　削除確認画面の表示（/character_delete/number）について

削除確認画面の表示（/character_delete/number）の機能の詳細についてプログラム 30 を参照しながら説明を行う．

プログラム 30　削除確認画面の表示（/character_delete/number）

```
1  app.get("/character_delete/:number", (req, res) => {
2      const number = req.params.number;
3      const detail = states[ number ];
4      res.render('character_delete', {id: number, data: detail} );
5  });
```

プログラム 30 では，データ削除を実行する前に内容を確認するための画面表示する機能をを持つ．１行目では，GET メソッドによる/character_delete/:number へのアクセスを入力として受け取り，２，３行目で指定された番号に対応するデータを取得した上で，４行目で，削除確認用ページを出力する．

4.10.8　削除の実行（/holy/delete/number）について

削除の実行（/character/delete/number）の機能の詳細についてプログラム 31 を参照しながら説明を行う．

プログラム 31　削除の実行（/character/delete/number)

```
1  app.get("/character/delete/:number", (req, res) => {
2      states.splice( req.params.number, 1 );
3      res.redirect('/character' );
4  });
```

プログラム 31 では，パスパラメータで指定されたデータを配列から削除する処理を行う．２行目で，受け取った番号の配列の内容を削除する．削除完了後は一覧表示ページへリダイレクトすることで，最新のデータ状態を利用者に提示する．