Webプログラミング レポート

千葉工業大学
情報変革科学部 情報工学科

学生番号 25G1098

氏　　名 長野　志哉

1. はじめに

本レポートは，Node.js を用いて構築された Web アプリケーションに関する仕様書である．今回は，「Orangestar 曲一覧」，「Arknights キャラ一覧」，「Overwatch キャラ一覧」の 3 つのサービスを作成した．ソースコードは以下のリンクより取得できる．

`https://github.com/Eval-fk2/webpro_report`

2. Orangestar曲一覧 利用者向け仕様書

1 サービス概要

Orangestar 曲一覧とは，ボカロ P，Orangestar 氏によって作られた多くの楽曲の情報をまとめたサービスである．

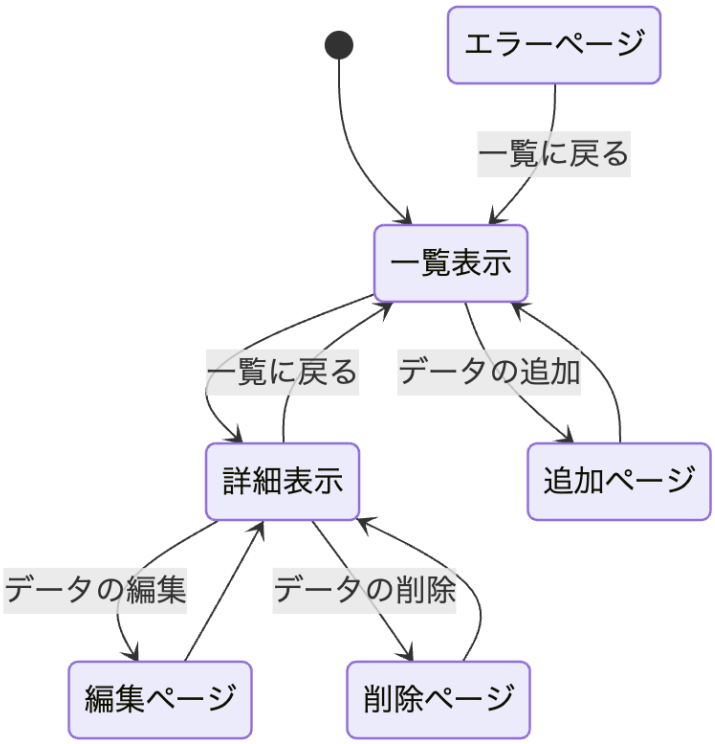


図 2.1 ページ遷移図

2 アクセス方法

`http://localhost:8080/db`
から，一覧表示ページにアクセスできる．

3 一覧表示ページ

Orangestar曲一覧

曲を追加する

ID	曲名	作曲	作詞	編曲	歌唱	URL
1	或る日と夏の追憶	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/6mFFtuHcfp8?si=nBCCnEfuCuvtaxE0
2	空奏列車	Orangestar	Orangestar	Orangestar	IA, 初音ミク	https://youtu.be/xzoShzMlllM?si=HbOLERbkt4zVJLU_
3	シンクロナイザー	Orangestar	Orangestar	Orangestar	初音ミク	https://youtu.be/gCxvFRuk4Xg?si=-hbT3JJ3-YoDjdun
4	真夏と少年の天ノ川戦争	Orangestar	Orangestar	Orangestar	IA	https://www.nicovideo.jp/watch/sm21293261
5	残灯花火	Orangestar	Orangestar	Orangestar	初音ミク	https://www.nicovideo.jp/watch/sm26786912
6	イヤホンと蝉時雨	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/UBqPPkB_Ybc?si=QIA07JaCVla1QDKy
7	超次元恋愛	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/zkpSBgRqs78?si=4Ls8WdEXb-IIA0xi
8	Ifの世界設定	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/Z-XXNvgI2Rg?si=t6l9ZLWaFm0P1OHL
9	からっぽの街月夜の下	Orangestar	Orangestar	Orangestar	初音ミク	https://youtu.be/1K00XxAw860?si=BlJ8cGsC9zw9SBV2
10	夏色アンサー	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/S6_zybtKN5k?si=HUuFwnRMwv7TMbQJ
11	白昼都市サブマージ計画	Orangestar		Orangestar		https://youtu.be/7edngkVJNak?si=XqP2Hj41sjb1H-JY
12	アスノヨゾラ哨戒班	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/XogSflwXgpw?si=zdN0kubCZM0bdARV
13	花と記憶	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/QdJaBpLqzQ4?si=TpRe8KCnyWZEV7mh
14	雨き声残響	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/0KK5vQlCVYo?si=L-vL1j0ehj0pY1ZH
15	未完成タイムリミッター	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/ISffFoK0Lv0?si=umpUpnB8UJx4kCuc
16	突風マニュアル	Orangestar	Orangestar	Orangestar	IA	https://www.nicovideo.jp/watch/sm21495069
17	東京廃都0区	Orangestar	Orangestar	Orangestar		
18	午前四時半	Orangestar	Orangestar	Orangestar	IA	https://www.nicovideo.jp/watch/sm23852034
19	CITRUS	Orangestar	Orangestar	Orangestar	IA, ONE	https://youtu.be/t8JXBtezzOc?si=Qondvmw37ZZD04_H
20	心象蜃気楼	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/VX9kZeHaklE?si=LVwnLu0olevWKw63
21	キミノヨゾラ哨戒班	Orangestar	Orangestar	Orangestar, 鈴木秋則	IA	https://youtu.be/QJNANqm9QPQ?si=8AuMQiPQOpgjNNoe
22	Lingering Fireworks	Orangestar	Orangestar, ColeenaWu, ★ham	Orangestar	初音ミク	https://www.nicovideo.jp/watch/sm26787011

図 2.2 一覧表示ページ

一覧表示ページでは，登録されている楽曲の情報を一覧で見ることができる．
下記の情報が表示される．

- id
- 曲名
- 作曲者
- 作詞者
- 編曲者
- 歌唱者
- URL

「曲を追加する」より，追加ページへアクセスする．
楽曲名をクリックすることで，各楽曲の詳細表示ページへアクセスする．
URL をクリックすることで，各楽曲を再生可能なページへアクセスする．

4 追加ページ

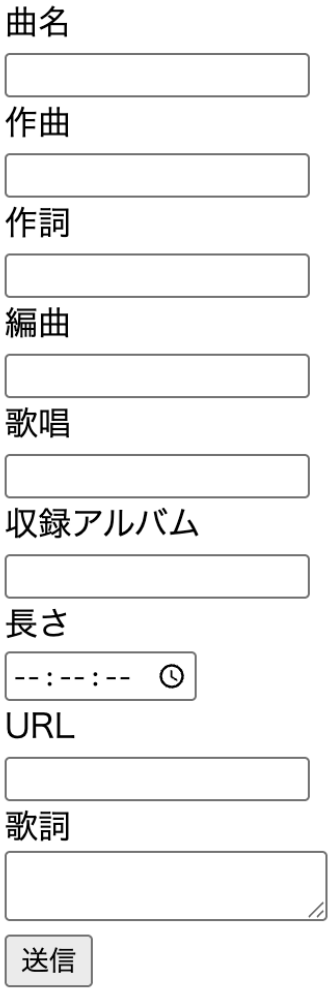


図 2.3 追加ページ

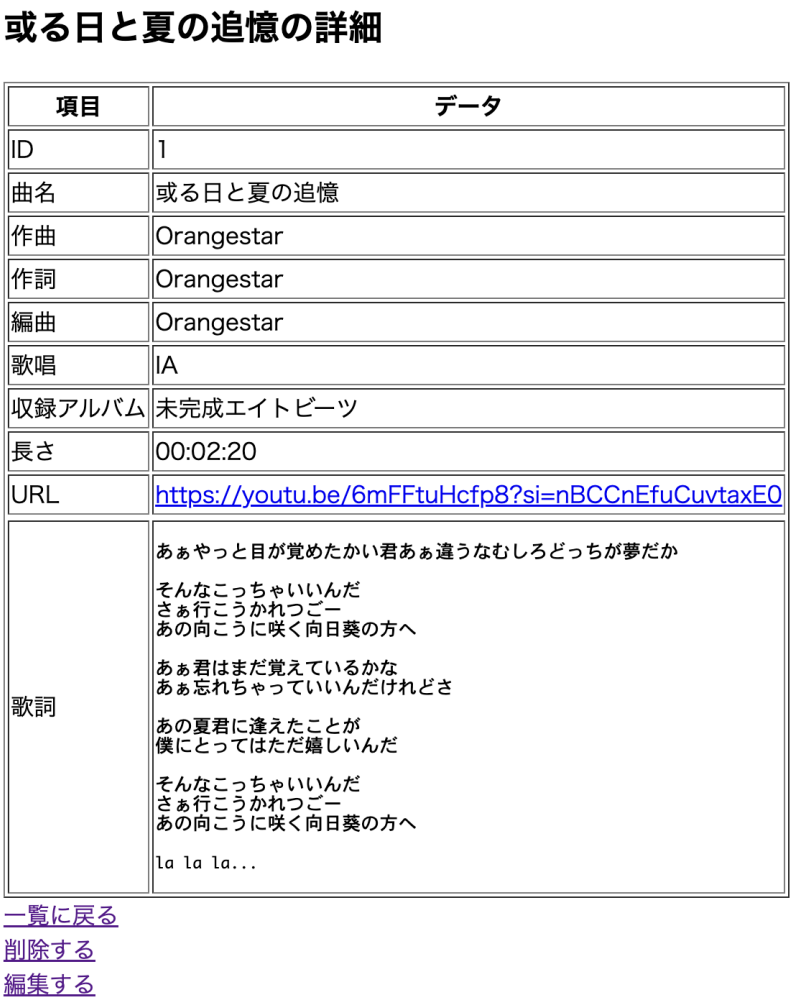
追加ページでは，新たに楽曲を追加できる．
下記の情報を入力できる．

- 曲名
- 作曲者
- 作詞者
- 編曲者
- 歌唱者
- 収録アルバム名

- 曲の長さ
- URL
- 歌詞

id は自動的に決定されるので，ユーザーは指定できない．
曲名のみ入力必須となっている．
送信ボタンを押すと，曲が追加され，一覧表示ページに移動する．

5 詳細表示ページ



或る日と夏の追憶の詳細

項目	データ
ID	1
曲名	或る日と夏の追憶
作曲	Orangestar
作詞	Orangestar
編曲	Orangestar
歌唱	IA
収録アルバム	未完成エイトビーツ
長さ	00:02:20
URL	https://youtu.be/6mFFtuHcfp8?si=nBCCnEfuCuvtaxE0
歌詞	あぁやっと目が覚めたかい君あぁ違うなむしろどっちが夢だか そんなこっちゃいいんだ さぁ行こうかれつごー あの向こうに咲く向日葵の方へ あぁ君はまだ覚えているかな あぁ忘れちゃっていいんだけれどさ あの夏君に逢えたことが 僕にとってはただ嬉しいんだ そんなこっちゃいいんだ さぁ行こうかれつごー あの向こうに咲く向日葵の方へ la la la...

一覧に戻る
削除する
編集する

図 2.4 詳細表示ページ

詳細表示ページでは，その楽曲が持つすべての情報を見ることができる．
下記の情報が表示される．

- id

- 曲名
- 作曲者
- 作詞者
- 編曲者
- 歌唱者
- 収録アルバム名
- 曲の長さ
- URL
- 歌詞

URL をクリックすることで，各楽曲を再生可能なページへアクセスする．
「一覧に戻る」より，一覧表示ページに戻る．
「削除する」より，削除ページへアクセスする．
「編集する」より，編集ページへアクセスする．

6 削除ページ

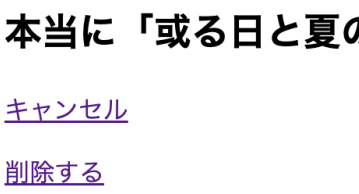


図 2.5 削除ページ

削除ページでは，曲を削除する際の最終確認が求められる．
「キャンセル」より，詳細表示ページに戻る．
「削除する」より，曲を削除し，一覧表示ページへアクセスする．

7 編集ページ

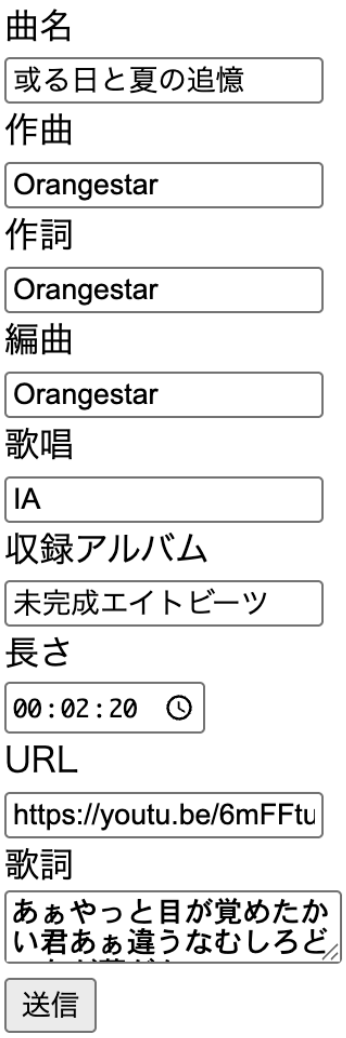


図 2.6 編集ページ

編集ページでは，既存の曲の情報を編集することができる．
下記の情報を編集できる．

- 曲名
- 作曲者
- 作詞者
- 編曲者
- 歌唱者
- 収録アルバム名

- 曲の長さ
- URL
- 歌詞

id は自動的に決定されるので，ユーザーは編集できない．
曲名のみ入力必須となっている．
送信ボタンを押すと，曲の情報が編集され，詳細表示ページに移動する．

8 エラーページ

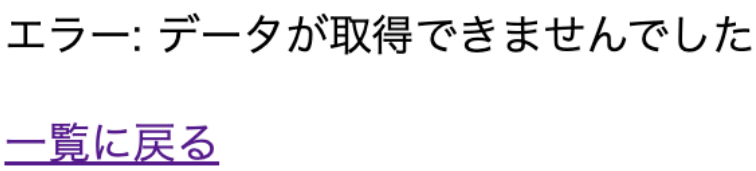


図 2.7 エラーページ

ページ間を移動する際，曲の情報がうまく取得できないとエラーとなり，このページが表示される．
「一覧に戻る」より，一覧表示ページに戻る．

3. 管理者向け仕様書

1 動作環境

本システムは Node.js を用いて構築された Web アプリケーションであり，以下の環境で動作する．

- OS：macOS
- Node.js：v20 以上
- Package Manager: npm
- Web ブラウザ：Google Chrome

2 必要なソフトウェアおよびインストール手順

2.1 Node.js のインストール

以下の公式サイトより Node.js をダウンロードし，インストールする．
URL：`https://nodejs.org/`

2.2 必要パッケージのインストール

ターミナルより，/webpro_report ディレクトリへ移動し，以下のコマンドを実行する．

```
npm install
```

3 フォルダ構成

本システムは複数のサービスごとにフォルダが分割されており，各サービスは
`project_{プロジェクト名}`
といった名称で管理される．各サービスを起動する際は該当フォルダへ移動してから実行する必要がある．

4 起動手順

4.1 サービスフォルダへ移動

実行したいサービス（例：project_serviceA）のフォルダに移動する.

```
cd project_serviceA
```

4.2 サーバの起動

以下のコマンドを実行することで，対象サービスの Web サーバが起動する.

```
node app.js
```

ターミナルに Example app listening on port 8080! と表示されていれば起動が成功している.

起動後，ブラウザで以下にアクセスする.

```
http://localhost:8080/db
```

4.3 サーバの停止

実行中のターミナルで以下の操作を行う.

```
Control + C
```

これにより Web サーバが停止する.

4. Orangestar曲一覧 開発者向け仕様書

1 サービス概要

Orangestar 曲一覧とは，ボカロ P，Orangestar 氏によって作られた多くの楽曲の情報をまとめたサービスである．

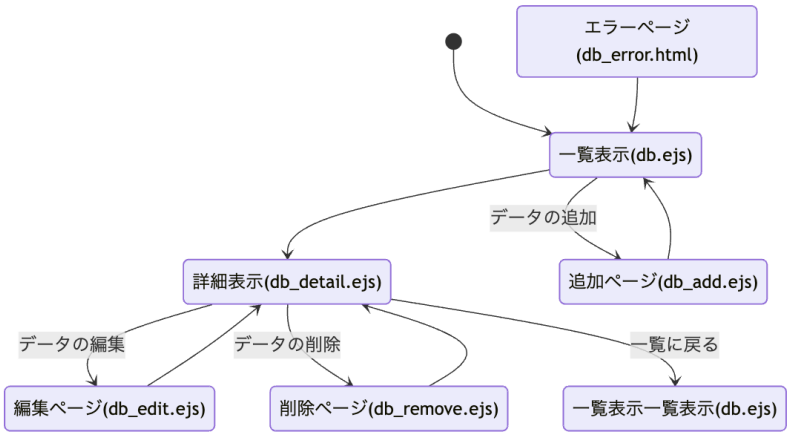


図 4.1 ページ遷移図

図 4.2 フォルダ構成

2 プログラムの説明

2.1 app.js

2.1.1 app.js の概要と全体

app.js は，データの管理やページ遷移など，このサービスにおけるメインの処理を務める．

```
1   "use strict";
2   const strict = require('assert/strict');
3   const express = require('express');
4   const app = express();
5
6   app.set('view engine', 'ejs');
7   app.use('/public', express.static(__dirname + '/public'));
8
9   let idManager = 0;
10  const properties = [
11    {propertyName: 'name', label: '曲名', type: 'text', required: true},
12    {propertyName: 'composer', label: '作曲', type: 'text'},
13    {propertyName: 'lyrics', label: '作詞', type: 'text'},
14    {propertyName: 'arangement', label: '編曲', type: 'text'},
15    {propertyName: 'vocal', label: '歌唱', type: 'text'},
16    {propertyName: 'album', label: '収録アルバム', type: 'text', view: false},
17    {propertyName: 'long', label: '長さ
          ', type: 'time', min: "00:00", step: 1, view: false},
18    {propertyName: 'url', label: 'URL', type: 'url'},
19    {propertyName: 'words', label: '歌詞', type: 'textarea', view: false},
20  ];
21  const propertySettings = {
22    main: 'name',
23    click: 'name'
24  };
25  let dataList = [];
26
27  setDefaultData();
28
29  app.get('/db', (req, res) => {
30    res.render('db', {dataList, properties, propertySettings} );
31  });
32
33  app.get('/db/:id', (req, res) => {
34    const id = req.params.id;
35    const data = dataList.find(a => a.id == id);
36    if (data === undefined) {
37      error(res);
38      return;
39    };
40    res.render('db_detail', {data, properties, propertySettings} );
41  });
42
43  app.get('/db_add', (req, res) => {
44    res.render('db_add', {properties});
45  });
46
47  app.get('/db_add_complete', (req, res) => {
```

```
48      const data = req.query;
49      addData(data);
50      res.redirect('/db');
51    });
52
53    app.get('/db_remove/:id', (req, res) => {
54      const id = req.params.id;
55      const data = dataList.find(a => a.id == id);
56      if (data === undefined) {
57        error(res);
58        return;
59      };
60      res.render('db_remove',{data, propertySettings});
61    });
62
63    app.get('/db_remove_complete/:id', (req, res) => {
64      const id = req.params.id;
65      removeData(id);
66      res.redirect('/db');
67    });
68
69    app.get('/db_edit/:id', (req, res) => {
70      const id = req.params.id;
71      const data = dataList.find(a => a.id == id);
72      if (data === undefined) {
73        error(res);
74        return;
75      };
76      res.render('db_edit', {data, properties});
77    });
78
79    app.get('/db_edit_complete/:id', (req, res) => {
80      const id = req.params.id;
81      const data = req.query;
82      const edit = editData(id, data);
83      if (edit === undefined) {
84        error(res);
85        return;
86      }
87      res.redirect(`/db/${id}`);
88    });
89
90    app.listen(8080, () => console.log('Example app listening on port 8080!'));
91
92    function getId() {
93      idManager++;
94      return idManager;
95    };
96
97    function addData(data) {
98      data['id'] = getId();
99      dataList.push(data);
100   };
101
102   function removeData(id) {
103     dataList = dataList.filter(a => a.id != id);
104   };
105
106   function editData(id, data) {
```

```
107       const dataIndex = dataList.findIndex(a => a.id == id);
108       if (dataIndex === -1) return undefined;
109       for (const {propertyName} of properties) {
110         dataList[dataIndex][propertyName] = data[propertyName];
111       };
112       return true;
113     };
114
115     function error(res) {
116       res.redirect('/public/db_error.html');
117     };
118
119     function setDefaultData() {
120       const defaultData = [
121         ['曲名', '作曲者', '作詞者', '編曲', '歌唱者', '収録アルバム
              ', '00:00:00', 'https://test', '歌詞'],
122         ['曲名', '作曲者', '作詞者', '編曲', '歌唱者', '収録アルバム
              ', '00:00:00', 'https://test', '歌詞'],
123         ['曲名', '作曲者', '作詞者', '編曲', '歌唱者', '収録アルバム
              ', '00:00:00', 'https://test', '歌詞'],
124         ...
125       ];
126       for (const data of defaultData) {
127         const fixedData = {};
128         for (let i = 0; i <properties.length; i++) {
129           fixedData[properties[i].propertyName] = data[i];
130         };
131         addData(fixedData);
132       };
133     };
```

2.1.2 使用する変数

- idManger(9 行目): 各曲のオブジェクトごとに割り当てる固有の id を管理する変数.
- properties(10 行目): 曲が持つ複数の情報を各項目ごとに管理，設定する定数.

表 4.1 properties 内のオブジェクトで使用可能な要素一覧

要素名	役割	必須/任意
propertyName	プログラム内での項目名を設定する.	必須
label	ブラウザ上で表示する項目名を設定する.	必須
type	項目の情報種別を設定する．また，これによって曲の追加や編集時の情報の入力方法が変化する．使用可能なタイプは表 6.2 に示す.	必須
required	その項目が入力必須項目かを設定する. undefined，または false で任意．true で必須.	任意
view	一覧表示ページで項目を表示するか設定する. undefined，または true で表示．false で非表示.	任意

表 4.2 type で使用可能なタイプ一覧

タイプ	役割	各タイプが持つ固有要素		
		要素名	役割	任意/必須
'text'	1 行のみの文字列の情報を扱う.	-	-	-
'number'	数値の情報を扱う.	min	数値のみ入力可能. 指定可能な最小値を設定する.	任意
		max	数値のみ入力可能. 指定可能な最大値を設定する.	任意
		step	数値のみ入力可能. 指定可能な数値の間隔を設定する.	任意
'select'	ドロップダウンで特定の選択肢内の情報を扱う.	options	配列のみ入力可能. 選択肢を設定する.	必須
'url'	URL の情報を扱う.	-	-	-
'time'	時間の情報を扱う.	min	時刻文字列('00:00:00') のみ入力可能. 指定可能な最小時間を設定する.	任意
		max	時刻文字列('00:00:00') のみ入力可能. 指定可能な最大時間を設定する.	任意
		step	数値のみ入力可能. 指定可能な時間(秒)の間隔を設定する.	任意
'textarea'	改行ありの文字列の情報を扱う.	-	-	-

- propertySettings(21 行目): サービス全体において，特定の役割を果たす項目を設定する定数.

表 4.3 propertySettings で使用可能な要素一覧

要素名	役割
main	項目名のみ入力可能. 曲のオブジェクトをブラウザ上でユーザーに向けて表す時，代表となる項目を設定する.
click	項目名のみ入力可能. 一覧表示ページから詳細表示ページに移動するためのクリック機能をどの項目に割り当てるのかを設定する.

- dataList: 曲のオブジェクトを保存しておくための変数.

2.1.3 ページ遷移機能

- '/db'(29 行目): 一覧表示ページ (db.ejs) を呼び出す機能を持つ.
  一覧表示ページ (db.ejs) に dataList，properties，propertysettings を渡して呼び出す.

- '/db/:id'(33 行目): 詳細表示ページ (db_detail.ejs) を呼び出す機能を持つ.
  url の id から曲のオブジェクトを定数 dta として取得し，詳細表示ページ (db_detail.ejs) に data，properties，propertySettings を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_add'(43 行目): 追加ページ (db_add.ejs) を呼び出す機能を持つ.
  追加ページ (db_add.ejs) に properties を渡して呼び出す.

- '/db_add_complete'(47 行目): 曲の追加処理を行う機能を持つ.
  追加ページ (db_add.ejs) から入力された情報を定数 data として受け取り，関数 addData に data を渡して追加処理をする.
  終了後は'/db' にリダイレクトする.

- '/db_remove/:id'(53 行目): 削除ページ (db_remove.ejs) を呼び出す機能を持つ.
  url の id から曲のオブジェクトを定数 data として取得し，削除ページ (db_remove.ejs) に data，propertySettings を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_remove_complete/:id'(63 行目): 曲の削除処理を行う機能を持つ.
  url の id を関数 removeData に渡して削除処理をする.
  終了後は'/db' にリダイレクトする.

- '/db_edit/:id'(69 行目): 編集ページ (db_edit.ejs) を呼び出す機能を持つ.
  url の id から曲のオブジェクトを定数 data として取得し，編集ページ (db_edit.ejs) に data，properties を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_edit_complete/:id'(79 行目): 曲の編集処理を行う機能を持つ.
  編集ページ (db_edit.ejs) から入力された情報を定数 data として受け取り，url の id と data を関数 editData に渡して編集処理をする.
  関数 editData から false を受け取った場合は関数 error を実行し，return する.
  終了後は'/db/:id' にリダイレクトする.

2.1.4 使用する関数

- getId(92 行目) 入力: なし，出力: int
  実行されるごとに id として固有の数値を返す機能を持つ.

idManager に 1 を足してその値を返す．

- addData(97 行目) 入力: Object data，出力: なし
  入力された新しい曲のオブジェクトに id を与えて dataList に保存する機能を持つ．
  関数 getId によって id を与え，dataList に push する．

- removeData(102 行目) 入力: int id，出力: なし
  入力された id をもとに dataList からオブジェクトを削除する機能を持つ．
  入力された id 一致しない id を持つオブジェクトのみで構成される配列を作成し，それを dataList に上書きする．

- editData(106 行目) 入力: int id，Object data，出力: boolean
  dataList 内の既存のオブジェクトの情報を新しいオブジェクトの情報に書き換える機能を持つ．
  入力された id より dataList から既存のオブジェクトの位置を探索し，入力された data より新しいオブジェクトの情報に上書きする．
  dataList 内にオブジェクトが見つからなかった場合は false を，そうでなければ true を返す．

- error(115 行目) 入力: Object res，出力: なし
  エラーページ ('db_error.html') を呼び出す機能を持つ．

- setDefaultData(119 行目) 入力: なし，出力: なし
  dataList に初期値としてオブジェクトを登録する機能を持つ．
  関数内の定数 defaultData は二次元配列になっており，外側の配列内の要素は一つのオブジェクトを表しており，内側の配列内の要素は properties の順番にオブジェクト内の要素として登録されていく．
  定数 fixedData としてオブジェクト化された各要素は fixedData を関数 addData に渡して dataList に追加される．

2.2 db.ejs

2.2.1 db.ejs の概要と全体

db.ejs は複数の曲情報の一覧を表示するためのページであり，各曲の id や設定された項目値を表形式で表示する機能を持つ．また，追加ページや詳細ページへの遷移リンクも提供する．

Orangestar曲一覧

曲を追加する

ID	曲名	作曲	作詞	編曲	歌唱	URL
1	或る日と夏の追憶	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/6mFFtuHcfp8?si=nBCCnEfuCuvtaxE0
2	空奏列車	Orangestar	Orangestar	Orangestar	IA, 初音ミク	https://youtu.be/xzoShzMlIlM?si=HbOLERbkt4zVJLU_
3	シンクロナイザー	Orangestar	Orangestar	Orangestar	初音ミク	https://youtu.be/gCxvFRuk4Xg?si=-hbT3JJ3-YoDjdun
4	真夏と少年の天ノ川戦争	Orangestar	Orangestar	Orangestar	IA	https://www.nicovideo.jp/watch/sm21293261
5	残灯花火	Orangestar	Orangestar	Orangestar	初音ミク	https://www.nicovideo.jp/watch/sm26786912
6	イヤホンと蝉時雨	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/UBqPPkB_Ybc?si=QlA07JaCVla1QDKy
7	超次元恋愛	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/zkpSBgRqs78?si=4Ls8WdEXb-llA0xi
8	Ifの世界設定	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/Z-XXNvgI2Rg?si=t6l9ZLWaFm0P1OHL
9	からっぽの街月夜の下	Orangestar	Orangestar	Orangestar	初音ミク	https://youtu.be/1K00XxAw860?si=BlJ8cGsC9zw9SBV2
10	夏色アンサー	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/S6_zybtKN5k?si=HUuFwnRMwv7TMbQJ
11	白昼都市サブマージ計画	Orangestar		Orangestar		https://youtu.be/7edngkVJNak?si=XqP2Hj41sjb1H-JY
12	アスノヨゾラ哨戒班	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/XogSflwXgpw?si=zdN0kubCZM0bdARV
13	花と記憶	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/QdJaBpLqzQ4?si=TpRe8KCnyWZEV7mh
14	雨き声残響	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/0KK5vQlCVYo?si=L-vL1j0ehj0pY1ZH
15	未完成タイムリミッター	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/ISffFoK0Lv0?si=umpUpnB8UJx4kCuc
16	突風マニュアル	Orangestar	Orangestar	Orangestar	IA	https://www.nicovideo.jp/watch/sm21495069
17	東京廃都0区	Orangestar	Orangestar	Orangestar		
18	午前四時半	Orangestar	Orangestar	Orangestar	IA	https://www.nicovideo.jp/watch/sm23852034
19	CITRUS	Orangestar	Orangestar	Orangestar	IA, ONE	https://youtu.be/t8JXBtezzOc?si=Qondvmw37ZZD04_H
20	心象蜃気楼	Orangestar	Orangestar	Orangestar	IA	https://youtu.be/VX9kZeHaklE?si=LVwnLu0olevWKw63
21	キミノヨゾラ哨戒班	Orangestar	Orangestar	Orangestar, 鈴木秋則	IA	https://youtu.be/QJNANqm9QPQ?si=8AuMQiPQQpgjNNoe
22	Lingering Fireworks	Orangestar	Orangestar, ColeenaWu, ✲ham	Orangestar	初音ミク	https://www.nicovideo.jp/watch/sm26787011

図 4.3 一覧表示ページ

```
<!Doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>曲一覧 Orangestar</title>
</head>
<body>
  <h2>曲一覧 Orangestar</h2>
  <p><a href="/db_add">曲を追加する</a></p>
  <table border="1">
    <tr>
      <th>ID</th>

      <% for (const property of properties) { %>
        <% if (property.view === undefined || property.view === true) { %><
            th><%= property.label %></th><% }; %>
      <% }; %>
    </tr>

    <% for (const data of dataList) { %>
      <tbody>
        <td><%= data.id %></td>
        <% for (const property of properties) { %>
          <% if (property.view !== undefined && property.view === false)
              continue; %>
          <% if (property.propertyName === propertySettings.click) { %>
            <td><a href="/db/<%= data.id %>"><%= data[property.propertyName]
                %></a></td>
          <% } else if (property.type === 'url') { %>
            <td><a href="<%= data[property.propertyName] %>"><%= data[
                property.propertyName] %></a></td>
          <% } else if (property.type === 'textarea') { %>
            <td><pre><%= data[property.propertyName] %></pre></td>
          <% } else { %>
            <td><%= data[property.propertyName] %></td>
```

```
32              <% }; %>
33            <% }; %>
34          </tbody>
35        <% }; %>
36      </table><br>
37    </body>
38    </html>
```

2.2.2 ページ遷移機能

- 追加ページへの遷移 (9 行目):「曲を追加する」リンクをクリックすると，追加ページ (db_add.ejs) へ移動する．これは app.js の'/db_add' に対応する．
- 詳細表示ページへの遷移: properties のうち，propertySettings.click に設定された項目は一覧表示時にリンクとして表示され，クリックすると詳細ページ (db_detail.ejs) へ移動する．app.js の'/db/:id' に対応する．

2.2.3 一覧表示機能

10 行目から table を作成し，一覧表示している．id は必ず表示する項目として 12 行目に直接記述し，その他の項目は 14 行目より properties の view をもとに表示/非表示を判別している．

19 行目よりオブジェクトの各項目の情報を view をもとに表示/非表示を判別している．各情報は以下の優先順位で表示形式が決定される．

- click: propertySettings の click に指定されている項目は詳細表示ページ (db_detail.ejs) に移動する機能を持ち，クリックすると app.js の'/db/:id' を呼び出す．
- url: タイプが url の場合は実際に url にアクセスできるようになっている．
- textare: タイプが textarea の場合は情報が¡pre¿タグで囲まれて表示されるようになっている．
- その他: 情報がそのまま表示される．

これらの表示形式は共存しない．

2.3 db_detail.ejs

db_detail.ejs は 1 曲の情報をより詳細に表示するためのページであり，一覧表示ページでの情報に加えて，収録アルバムや，曲の長さを表形式で表示する機能を持つ．

或る日と夏の追憶の詳細

項目	データ
ID	1
曲名	或る日と夏の追憶
作曲	Orangestar
作詞	Orangestar
編曲	Orangestar
歌唱	IA
収録アルバム	未完成エイトビーツ
長さ	00:02:20
URL	https://youtu.be/6mFFtuHcfp8?si=nBCCnEfuCuvtaxE0
歌詞	あぁやっと目が覚めたかい君あぁ違うなむしろどっちが夢だか そんなこっちゃいいんだ さぁ行こうかれつごー あの向こうに咲く向日葵の方へ あぁ君はまだ覚えているかな あぁ忘れちゃっていいんだけれどさ あの夏君に逢えたことが 僕にとってはただ嬉しいんだ そんなこっちゃいいんだ さぁ行こうかれつごー あの向こうに咲く向日葵の方へ la la la...

一覧に戻る
削除する
編集する

図 4.4 詳細表示ページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title><%= data[propertySettings.main] %></title>
6   </head>
7   <body>
8     <h2><%= data[propertySettings.main] %>の詳細</h2>
9     <table border="1">
10      <tr><th>項目</th><th>データ</th></tr>
11      <tr><td>ID</td><td><%= data.id %></td></tr>
12      <% for (const property of properties ) { %>
13        <% if (property.type === 'url') { %>
14          <tr><td><%= property.label %></td><td><a href="<%= data[property.
                propertyName] %>"><%= data[property.propertyName] %></a></td><tr
                >
15        <% } else if (property.type === 'textarea') { %>
16          <tr><td><%= property.label %></td><td><pre><%= data[property.
                propertyName] %></pre></td></tr>
17        <% } else { %>
18          <tr><td><%= property.label %></td><td><%= data[property.propertyName
                ] %></td></tr>
```

```
19          <% }; %>
20        <% }; %>
21      </table>
22      <a href="/db">一覧に戻る</a><br>
23      <a href="/db_remove/<%= data.id%>">削除する</a><br>
24      <a href="/db_edit/<%= data.id%>">編集する</a><br>
25    </body>
26    </html>
```

2.3.1 タイトル変更機能

5 行目ではタイトルを，8 行目では見出しを propertySettings.main で指定された項目の値を表示するようにしている．

2.3.2 ページ遷移機能

- 一覧表示ページへの遷移 (22 行目):「一覧に戻る」リンクをクリックすると，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db’ に対応する．
- 削除ページへの遷移 (23 行目):「削除する」リンクをクリックすると，削除ページ (db_remove.ejs) へ移動する．これは app.js の’/db_remove’ に対応する．
- 編集ページへの遷移 (24 行目):「編集する」リンクをクリックすると，編集ページ (db_edit.ejs) へ移動する．これは app.js の’/db_edit’ に対応する．

2.3.3 詳細表示機能

9 行目から table を作成し，詳細表示している．id は必ず表示する項目として 11 行目に直接記述し，その他の項目は 12 行目より properties をもとに表示している．
各情報は以下の優先順位で表示形式が決定される．

- url: タイプが url の場合は実際に url にアクセスできるようになっている．
- textare: タイプが textarea の場合は情報が¡pre¿タグで囲まれて表示されるようになっている．
- その他: 情報がそのまま表示される．

これらの表示形式は共存しない．

2.4 db_add.ejs

2.4.1 db_add.ejs の概要と全体

db_add.ejs は新たな曲を追加するためのページであり，曲の各項目ごとの情報を入力できる機能を持つ.

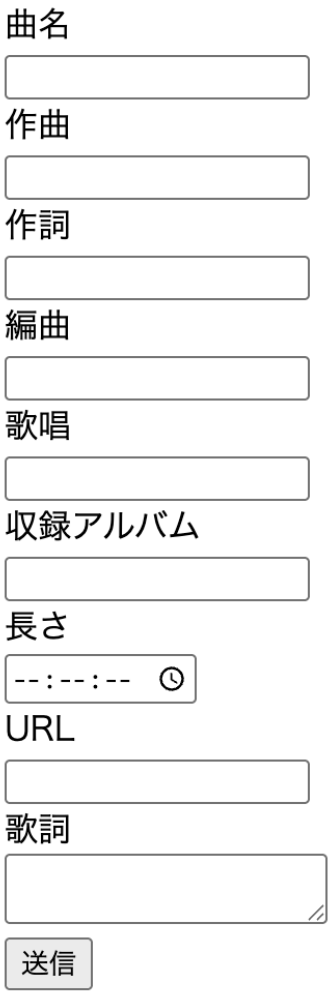


図 4.5 追加ページ

```
1    <!DOCTYPE html>
2    <html lang="ja">
3    <head>
4      <meta charset="UTF-8">
5      <title>追加</title>
6    </head>
7    <body>
8      <form action="/db_add_complete">
9        <% for (const property of properties) { %>
10         <% if (property.type === 'text') { %>
11           <label for="<%= property.propertyName %>"><%= property.label%></
                 label><br>
12           <input type="<%= property.type %>" name="<%= property.propertyName %
                 >" id="<%= property.propertyName %>"
```

```
13            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
14            ><br>
15          <% }; %>
16
17          <% if (property.type === 'number') { %>
18            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
19            <input
20              type="number"
21              name="<%= property.propertyName %>"
22              id="<%= property.propertyName %>"
23              <% if (property.min !== undefined) { %> min="<%= property.min %>"
                    <% }; %>
24              <% if (property.max !== undefined) { %> max="<%= property.max %>"
                    <% }; %>
25              <% if (property.step !== undefined) { %> step="<%= property.step %
                    >" <% }; %>
26              <% if (property.required !== undefined && property.required ===
                    true) { %>required<% }; %>
27            ><br>
28          <% } %>
29
30          <% if (property.type === 'select') { %>
31            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
32            <select id="<%= property.propertyName %>" name="<%= property.
                  propertyName %>">
33              <% for (const optionData of property.options) { %>
34                <option value="<%= optionData %>"><%= optionData %></option>
35              <% }; %>
36            </select><br>
37          <% }; %>
38
39          <% if (property.type === 'url') { %>
40            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
41            <input type="<%= property.type %>" name="<%= property.propertyName %
                  >" id="<%= property.propertyName %>"
42            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
43            ><br>
44          <% }; %>
45
46          <% if (property.type === 'time') {%>
47            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
48            <input type="<%= property.type %>" name="<%= property.propertyName %
                  >" id="<%= property.propertyName %>"
49            <% if (property.min !== undefined) { %> min="<%= property.min %>" <%
                   }; %>
50            <% if (property.max !== undefined) { %> max="<%= property.max %>" <%
                   }; %>
51            <% if (property.step !== undefined) { %> step="<%= property.step %>"
                   <% }; %>
52            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
53            ><br>
54          <% }; %>
```

```
55
56          <% if (property.type === 'textarea') { %>
57            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
58            <textarea name="<%= property.propertyName %>" id="<%= property.
                  propertyName %>"
59            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
60            ></textarea><br>
61          <% }; %>
62        <% }; %>
63        <input type="submit">
64      </form>
65    </body>
66    </html>
```

2.4.2 情報入力機能

8 行目よりフォームを作成し，オブジェクトのタイプごとに入力欄を生成している．タイプごとに生成される入力欄は以下のとおりである．

- 'text'(10 行目): テキスト入力用の`<input type="text">`を生成する．
- 'number'(17 行目): 数値入力用の`<input type="number">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'select'(30 行目): 選択式入力のための`<select>`を生成する．選択肢は options より参照する．
- 'url'(39 行目): URL 入力用の`<input type="url">`を生成する．
- 'time'(46 行目): 時刻入力用の`<input type="time">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'textarea'(56 行目): 複数行テキスト入力用の`<textarea>`を生成する．

'select' タイプを除く各項目において required が true であれば required 属性を付与する．

2.4.3 情報送信機能

63 行目では送信ボタンとして`<input type="submit">`を生成し，app.js の'/db_add_complete' に入力した情報を送信する．

2.5 db_edit.ejs

2.5.1 db_edit.ejs の概要と全体

db_edit.ejs は曲の情報を編集するためのページであり，曲の各項目ごとの情報を入力できる機能を持つ.

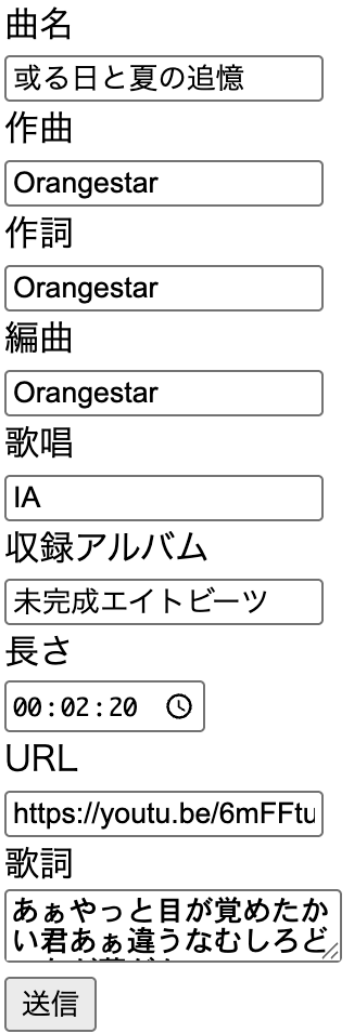


図 4.6 編集ページ

```
1   <!DOCTYPE html>
2   <html lang="ja">
3   <head>
4     <meta charset="UTF-8">
5     <title>編集</title>
6   </head>
7   <body>
8     <form action="/db_edit_complete/<%= data.id%>">
9       <% for (const property of properties) { %>
10        <% if (property.type === 'text') { %>
11          <label for="<%= property.propertyName %>"><%= property.label%></
              label><br>
```

```
12          <input type="<%= property.type %>" name="<%= property.propertyName %>" id="<%= property.propertyName %>" value="<%= data[property.propertyName]%>"
13          <% if (property.required !== undefined && property.required === true) { %>required<% }; %>
14          ><br>
15        <% }; %>
16
17        <% if (property.type === 'number') { %>
18          <label for="<%= property.propertyName %>"><%= property.label %></label><br>
19          <input
20            type="number"
21            name="<%= property.propertyName %>"
22            id="<%= property.propertyName %>"
23            value="<%= data[property.propertyName]%>"
24            <% if (property.min !== undefined) { %> min="<%= property.min %>" <% }; %>
25            <% if (property.max !== undefined) { %> max="<%= property.max %>" <% }; %>
26            <% if (property.step !== undefined) { %> step="<%= property.step %>" <% }; %>
27            <% if (property.required !== undefined && property.required === true) { %>required<% }; %>
28          ><br>
29        <% } %>
30
31        <% if (property.type === 'select') { %>
32          <label for="<%= property.propertyName %>"><%= property.label %></label><br>
33          <select id="<%= property.propertyName %>" name="<%= property.propertyName %>">
34            <% for (const optionData of property.options) { %>
35              <option value="<%= optionData %>"
36                <% if (data[property.propertyName] === optionData) { %> selected<% }; %>
37              ><%= optionData %></option>
38            <% }; %>
39          </select>
40        <% }; %>
41
42        <% if (property.type === 'url') { %>
43          <label for="<%= property.propertyName %>"><%= property.label %></label><br>
44          <input type="<%= property.type %>" name="<%= property.propertyName %>" id="<%= property.propertyName %>" value="<%= data[property.propertyName] %>"
45          <% if (property.required !== undefined && property.required === true) { %>required<% }; %>
46          ><br>
47        <% }; %>
48
49        <% if (property.type === 'time') {%>
50        <label for="<%= property.propertyName %>"><%= property.label %></label><br>
51        <input type="<%= property.type %>" name="<%= property.propertyName %>" id="<%= property.propertyName %>" value="<%= data[property.propertyName] %>"
52        <% if (property.min !== undefined) { %> min="<%= property.min %>" <%
```

```
            }; %>
53          <% if (property.max !== undefined) { %> max="<%= property.max %>" <%
                }; %>
54          <% if (property.step !== undefined) { %> step="<%= property.step %>" <
                % }; %>
55          <% if (property.required !== undefined && property.required === true)
                { %>required<% }; %>
56          ><br>
57          <% }; %>
58
59          <% if (property.type === 'textarea') { %>
60            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
61            <textarea name="<%= property.propertyName %>" id="<%= property.
                  propertyName %>"
62            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
63            ><%= data[property.propertyName]%></textarea><br>
64          <% }; %>
65        <% }; %>
66        <input type="submit">
67      </form>
68    </body>
69    </html>
```

2.5.2 情報入力機能

8 行目よりフォームを作成し，オブジェクトのタイプごとに入力欄を生成している．タイプごとに生成される入力欄は以下のとおりである．

- 'text'(10 行目): テキスト入力用の`<input type="text">`を生成する．
- 'number'(17 行目): 数値入力用の`<input type="number">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'select'(31 行目): 選択式入力のための`<select>`を生成する．選択肢は options より参照する．
- 'url'(42 行目): URL 入力用の`<input type="url">`を生成する．
- 'time'(49 行目): 時刻入力用の`<input type="time">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'textarea'(59 行目): 複数行テキスト入力用の`<textarea>`を生成する．

'select' タイプを除く各項目において required が true であれば required 属性を付与する．さらに，編集前のオブジェクトの値が value="..." や`<textarea>...</textarea>`として各入力欄の初期値に設定される．

2.5.3 情報送信機能

66 行目では送信ボタンとして<input type="submit">を生成し，app.js の’/db_edit_complete/:id’に入力した情報を送信する．

2.6 db_remove.ejs

2.6.1 db_remove.ejs の全体と概要

db_remove.ejs は曲を削除する際の最終確認をするページであり，曲を削除するか選択できる機能を持つ．

図 4.7 削除ページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>削除</title>
6   </head>
7   <body>
8     <h2>本当に「<%= data[propertySettings.main]%>」を削除しますか？ </h2>
9     <p><a href="/db/<%= data.id%>">キャンセル</a></p>
10    <p><a href="/db_remove_complete/<%= data.id%>">削除する</a></p>
11  </body>
12  </html>
```

2.6.2 オブジェクト名表示機能

8 行目では，プログラム上では曲をオブジェクトとして認識しているので，ユーザーが理解できるように propertySettings.main で指定された項目の値を表示するようにしている．

2.6.3 ページ遷移機能

- 詳細表示ページへの遷移 (9 行目):「キャンセル」リンクをクリックすると，詳細表示ページ (db_detail.ejs) へ移動する．これは app.js の’/db/:id’ に対応する．
- 削除処理の実行 (10 行目):「削除する」リンクをクリックすると，削除処理が実行さ

れ，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db_remove_complete’ に対応する．

2.7 db_error.html

2.7.1 db_error.html の概要と全体

db_error.ejs はオブジェクトが取得できなかった時に表示されるページであり，エラー動作から通常の動作へ戻す機能を持つ．

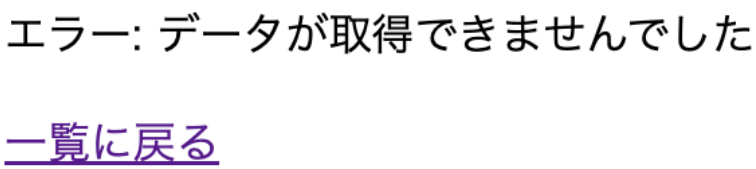


図 4.8 エラーページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>エラー</title>
6   </head>
7   <body>
8     <p>エラー: データが取得できませんでした</p>
9     <p><a href="/db">一覧に戻る</a></p>
10  </body>
11  </html>
```

2.7.2 ページ遷移機能

一覧表示ページへの遷移 (9 行目): 「一覧に戻る」リンクをクリックすると，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db’ に対応する．

5. Arknightsキャラ一覧 開発者向け仕様書

1 サービス概要

Arknights キャラ一覧とは，スマホゲームアークナイツに登場する多くのキャラクターの情報をまとめたサービスである．

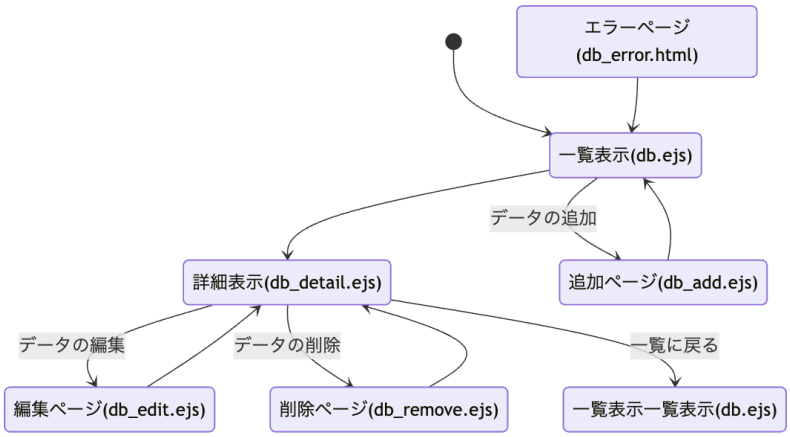


図 5.1 ページ遷移図

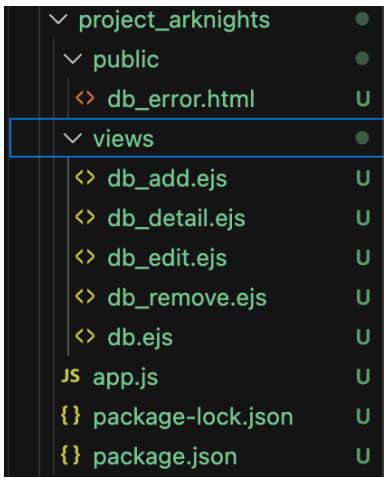


図 5.2 フォルダ構成

2 プログラムの説明

2.1 app.js

2.1.1 app.js の概要と全体

app.js は，データの管理やページ遷移など，このサービスにおけるメインの処理を務める．

```
1   "use strict";
2   const strict = require('assert/strict');
3   const express = require('express');
4   const app = express();
5
6   app.set('view engine', 'ejs');
7   app.use('/public', express.static(__dirname + '/public'));
8
9   let idManager = 0;
10  const properties = [
11    {propertyName: 'name', label: '名前', type: 'text', required: true},
12    {propertyName: 'rarity', label: 'レアリティ
        ', type: 'select', options: ['星1', '星2', '星3', '星4', '星5', '星
        6'], required: true},
13    {propertyName: 'role', label: '職業', type: 'select', options: ['先鋒
        ', '狙撃', '前衛', '術師', '重装', '医療', '特殊', '補助']},
14    {propertyName: 'hp', label: 'HP', type: 'number', min: 0, view: false,
        required: true},
15    {propertyName: 'atk', label: '攻撃力
        ', type: 'number', min: 0, step: 1, view: false, required: true},
16    {propertyName: 'def', label: '防御力
        ', type: 'number', min: 0, step: 1, view: false, required: true},
17    {propertyName: 'res', label: '術耐性
        ', type: 'number', min: 0, step: 1, view: false, required: true},
18    {propertyName: 'redep_time', label: '再配置
        ', type: 'number', min: 0, view: false, required: true},
19    {propertyName: 'cost', label: 'コスト
        ', type: 'number', min: 0, step: 1, view: false, required: true},
20    {propertyName: 'block', label: 'ブロック数
        ', type: 'number', min: 0, step: 1, view: false, required: true},
21    {propertyName: 'atk_int', label: '攻撃速度
        ', type: 'number', min: 0, view: false, required: true},
22    {propertyName: 'passive1', label: '素質
        1', type: 'textarea', view: false},
23    {propertyName: 'passive2', label: '素質
        2', type: 'textarea', view: false},
24    {propertyName: 'skill1', label: 'スキル
        1', type: 'textarea', view: false},
25    {propertyName: 'skill2', label: 'スキル
        2', type: 'textarea', view: false},
26    {propertyName: 'skill3', label: 'スキル3', type: 'textarea', view: false}
27  ];
28  const propertySettings = {
29    main: 'name',
30    click: 'name'
31  };
32  let dataList = [];
33
```

```
34    setDefaultData();
35
36    app.get('/db', (req, res) => {
37      res.render('db', {dataList, properties, propertySettings} );
38    });
39
40    app.get('/db/:id', (req, res) => {
41      const id = req.params.id;
42      const data = dataList.find(a => a.id == id);
43      if (data === undefined) {
44        error(res);
45        return;
46      };
47      res.render('db_detail', {data, properties, propertySettings} );
48    });
49
50    app.get('/db_add', (req, res) => {
51      res.render('db_add', {properties});
52    });
53
54    app.get('/db_add_complete', (req, res) => {
55      const data = req.query;
56      addData(data);
57      res.redirect('/db');
58    });
59
60    app.get('/db_remove/:id', (req, res) => {
61      const id = req.params.id;
62      const data = dataList.find(a => a.id == id);
63      if (data === undefined) {
64        error(res);
65        return;
66      };
67      res.render('db_remove',{data, propertySettings});
68    });
69
70    app.get('/db_remove_complete/:id', (req, res) => {
71      const id = req.params.id;
72      removeData(id);
73      res.redirect('/db');
74    });
75
76    app.get('/db_edit/:id', (req, res) => {
77      const id = req.params.id;
78      const data = dataList.find(a => a.id == id);
79      if (data === undefined) {
80        error(res);
81        return;
82      };
83      res.render('db_edit', {data, properties});
84    });
85
86    app.get('/db_edit_complete/:id', (req, res) => {
87      const id = req.params.id;
88      const data = req.query;
89      const edit = editData(id, data);
90      if (edit === false) {
91        error(res);
92        return;
```

```
93        }
94        res.redirect(`/db/${id}`);
95      });
96
97      app.listen(8080, () => console.log('Example app listening on port 8080!'));
98
99      function getId() {
100       idManager++;
101       return idManager;
102     };
103
104     function addData(data) {
105       data['id'] = getId();
106       dataList.push(data);
107     };
108
109     function removeData(id) {
110       dataList = dataList.filter(a => a.id != id);
111     };
112
113     function editData(id, data) {
114       const dataIndex = dataList.findIndex(a => a.id == id);
115       if (dataIndex === -1) return false;
116       for (const {propertyName} of properties) {
117         dataList[dataIndex][propertyName] = data[propertyName];
118       };
119       return true;
120     };
121
122     function error(res) {
123       res.redirect('/public/db_error.html');
124     };
125
126     function setDefaultData() {
127       const defaultData = [
128         ['キャラ名', 'レアリティ', '職業', 'HP', '攻撃力', '防御力', '術耐性', '再
              配置', 'コスト', 'ブロック数', '攻撃速度', '素質1', '素質2', 'スキル
              1', 'スキル2', 'スキル3'],
129         ['キャラ名', 'レアリティ', '職業', 'HP', '攻撃力', '防御力', '術耐性', '再
              配置', 'コスト', 'ブロック数', '攻撃速度', '素質1', '素質2', 'スキル
              1', 'スキル2', 'スキル3'],
130         ['キャラ名', 'レアリティ', '職業', 'HP', '攻撃力', '防御力', '術耐性', '再
              配置', 'コスト', 'ブロック数', '攻撃速度', '素質1', '素質2', 'スキル
              1', 'スキル2', 'スキル3'],
131         ...
132       ];
133       for (const data of defaultData) {
134         const fixedData = {};
135         for (let i = 0; i <properties.length; i++) {
136           fixedData[properties[i].propertyName] = data[i];
137         };
138         addData(fixedData);
139       };
140     };
```

2.1.2 使用する変数

- idManger(9 行目): 各キャラのオブジェクトごとに割り当てる固有の id を管理する変数.
- properties(10 行目): キャラが持つ複数の情報を各項目ごとに管理，設定する定数.

表 5.1 properties 内のオブジェクトで使用可能な要素一覧

要素名	役割	必須/任意
propertyName	プログラム内での項目名を設定する.	必須
label	ブラウザ上で表示する項目名を設定する.	必須
type	項目の情報種別を設定する．また，これによってキャラの追加や編集時の情報の入力方法が変化する．使用可能なタイプは表 6.2 に示す.	必須
required	その項目が入力必須項目かを設定する. undefined，または false で任意．true で必須.	任意
view	一覧表示ページで項目を表示するか設定する. undefined，または true で表示．false で非表示.	任意

表 5.2 type で使用可能なタイプ一覧

タイプ	役割	各タイプが持つ固有要素		
		要素名	役割	任意/必須
'text'	1 行のみの文字列の情報を扱う.	-	-	-
'number'	数値の情報を扱う.	min	数値のみ入力可能. 指定可能な最小値を設定する.	任意
		max	数値のみ入力可能. 指定可能な最大値を設定する.	任意
		step	数値のみ入力可能. 指定可能な数値の間隔を設定する.	任意
'select'	ドロップダウンで特定の選択肢内の情報を扱う.	options	配列のみ入力可能. 選択肢を設定する.	必須
'url'	URL の情報を扱う.	-	-	-
'time'	時間の情報を扱う.	min	時刻文字列('00:00:00') のみ入力可能. 指定可能な最小時間を設定する.	任意
		max	時刻文字列('00:00:00') のみ入力可能. 指定可能な最大時間を設定する.	任意
		step	数値のみ入力可能. 指定可能な時間(秒)の間隔を設定する.	任意
'textarea'	改行ありの文字列の情報を扱う.	-	-	-

- propertySettings(28 行目): サービス全体において，特定の役割を果たす項目を設定する定数.

表 5.3 propertySettings で使用可能な要素一覧

要素名	役割
main	項目名のみ入力可能. キャラのオブジェクトをブラウザ上でユーザーに向けて表す時，代表となる項目を設定する.
click	項目名のみ入力可能. 一覧表示ページから詳細表示ページに移動するためのクリック機能をどの項目に割り当てるのかを設定する.

- dataList: キャラのオブジェクトを保存しておくための変数.

2.1.3 ページ遷移機能

- '/db'(36 行目): 一覧表示ページ (db.ejs) を呼び出す機能を持つ.
  一覧表示ページ (db.ejs) に dataList，properties，propertysettings を渡して呼び出す.

- '/db/:id'(40 行目): 詳細表示ページ (db_detail.ejs) を呼び出す機能を持つ.
  url の id からキャラのオブジェクトを定数 dta として取得し, 詳細表示ページ (db_detail.ejs) に data，properties，propertySettings を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_add'(50 行目): 追加ページ (db_add.ejs) を呼び出す機能を持つ.
  追加ページ (db_add.ejs) に properties を渡して呼び出す.

- '/db_add_complete'(54 行目): キャラの追加処理を行う機能を持つ.
  追加ページ (db_add.ejs) から入力された情報を定数 data として受け取り，関数 addData に data を渡して追加処理をする.
  終了後は'/db' にリダイレクトする.

- '/db_remove/:id'(60 行目): 削除ページ (db_remove.ejs) を呼び出す機能を持つ.
  url の id からキャラのオブジェクトを定数 data として取得し, 削除ページ (db_remove.ejs) に data，propertySettings を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_remove_complete/:id'(70 行目): キャラの削除処理を行う機能を持つ.
  url の id を関数 removeData に渡して削除処理をする.
  終了後は'/db' にリダイレクトする.

- '/db_edit/:id'(76 行目): 編集ページ (db_edit.ejs) を呼び出す機能を持つ.
  url の id からキャラのオブジェクトを定数 data として取得し, 編集ページ (db_edit.ejs) に data，properties を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_edit_complete/:id'(86 行目): キャラの編集処理を行う機能を持つ.
  編集ページ (db_edit.ejs) から入力された情報を定数 data として受け取り，url の id と data を関数 editData に渡して編集処理をする.
  関数 editData から false を受け取った場合は関数 error を実行し，return する.
  終了後は'/db/:id' にリダイレクトする.

2.1.4 使用する関数

- getId(99 行目) 入力: なし，出力: int
  実行されるごとに id として固有の数値を返す機能を持つ.

idManager に 1 を足してその値を返す．

- addData(104 行目) 入力: Object data，出力: なし
  入力された新しいキャラのオブジェクトに id を与えて dataList に保存する機能を持つ．
  関数 getId によって id を与え，dataList に push する．

- removeData(109 行目) 入力: int id，出力: なし
  入力された id をもとに dataList からオブジェクトを削除する機能を持つ．
  入力された id 一致しない id を持つオブジェクトのみで構成される配列を作成し，それを dataList に上書きする．

- editData(113 行目) 入力: int id，Object data，出力: boolean
  dataList 内の既存のオブジェクトの情報を新しいオブジェクトの情報に書き換える機能を持つ．
  入力された id より dataList から既存のオブジェクトの位置を探索し，入力された data より新しいオブジェクトの情報に上書きする．
  dataList 内にオブジェクトが見つからなかった場合は false を，そうでなければ true を返す．

- error(122 行目) 入力: Object res，出力: なし
  エラーページ ('db_error.html') を呼び出す機能を持つ．

- setDefaultData(126 行目) 入力: なし，出力: なし
  dataList に初期値としてオブジェクトを登録する機能を持つ．
  関数内の定数 defaultData は二次元配列になっており，外側の配列内の要素は一つのオブジェクトを表しており，内側の配列内の要素は properties の順番にオブジェクト内の要素として登録されていく．
  定数 fixedData としてオブジェクト化された各要素は fixedData を関数 addData に渡して dataList に追加される．

2.2 db.ejs

2.2.1 db.ejs の概要と全体

db.ejs は複数のキャラの情報の一覧を表示するためのページであり，各キャラの id や設定された項目値を表形式で表示する機能を持つ．また，追加ページや詳細ページへの遷移リンクも提供する．

Arknightsキャラ一覧

キャラを追加する

ID	名前	レアリティ	職業
1	Mon3tr	星6	医療
2	アスカロン	星6	特殊
3	アルケット	星6	狙撃
4	イネス	星6	先鋒
5	イフリータ	星6	術師
6	ウィシャデル	星6	狙撃
7	ウルピアヌス	星6	前衛
8	エイヤフィヤトラ	星6	術師
9	エクシア	星6	狙撃
10	グレイディーア	星6	特殊
11	ケルシー	星6	医療
12	ゴールデングロー	星6	術師
13	サリア	星6	重装
14	シュウ	星6	重装
15	シルバーアッシュ	星6	前衛
16	シー	星6	術師
17	スカジ	星6	前衛
18	スズラン	星6	補助
19	スルト	星6	前衛
20	ソーンズ	星6	前衛
21	チェン	星6	前衛
22	チョンユエ	星6	前衛

図 5.3 一覧表示ページ

```
<!Doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>キャラ一覧 Arknights</title>
</head>
<body>
  <h2>キャラ一覧 Arknights</h2>
  <p><a href="/db_add">キャラを追加する</a></p>
  <table border="1">
    <tr>
```

```
12          <th>ID</th>
13
14          <% for (const property of properties) { %>
15            <% if (property.view === undefined || property.view === true) { %><
                th><%= property.label %></th><% }; %>
16          <% }; %>
17        </tr>
18
19        <% for (const data of dataList) { %>
20          <tbody>
21            <td><%= data.id %></td>
22            <% for (const property of properties) { %>
23              <% if (property.view !== undefined && property.view === false)
                    continue; %>
24              <% if (property.propertyName === propertySettings.click) { %>
25                <td><a href="/db/<%= data.id %>"><%= data[property.propertyName]
                      %></a></td>
26              <% } else if (property.type === 'url') { %>
27                <td><a href="<%= data[property.propertyName] %>"><%= data[
                      property.propertyName] %></a></td>
28              <% } else if (property.type === 'textarea') { %>
29                <td><pre><%= data[property.propertyName] %></pre></td>
30              <% } else { %>
31                <td><%= data[property.propertyName] %></td>
32              <% }; %>
33            <% }; %>
34          </tbody>
35        <% }; %>
36      </table><br>
37    </body>
38    </html>
```

2.2.2 ページ遷移機能

- 追加ページへの遷移 (9 行目):「キャラを追加する」リンクをクリックすると，追加ページ (db_add.ejs) へ移動する．これは app.js の'/db_add' に対応する．
- 詳細表示ページへの遷移: properties のうち，propertySettings.click に設定された項目は一覧表示時にリンクとして表示され，クリックすると詳細ページ (db_detail.ejs) へ移動する．app.js の'/db/:id' に対応する．

2.2.3 一覧表示機能

10 行目から table を作成し，一覧表示している．id は必ず表示する項目として 12 行目に直接記述し，その他の項目は 14 行目より properties の view をもとに表示/非表示を判別している．

19 行目よりオブジェクトの各項目の情報を view をもとに表示/非表示を判別している．各情報は以下の優先順位で表示形式が決定される．

- click: propertySettings の click に指定されている項目は詳細表示ページ (db_detail.ejs)

に移動する機能を持ち，クリックすると app.js の’/db/:id’ を呼び出す．

- url: タイプが url の場合は実際に url にアクセスできるようになっている．
- textare: タイプが textarea の場合は情報が¡pre¿タグで囲まれて表示されるようになっている．
- その他: 情報がそのまま表示される．

これらの表示形式は共存しない．

2.3 db_detail.ejs

db_detail.ejs は 1 キャラの情報をより詳細に表示するためのページであり，一覧表示ページでの情報に加えて，収録アルバムや，キャラの長さを表形式で表示する機能を持つ．

Mon3trの詳細

項目	データ
ID	1
名前	Mon3tr
レアリティ	星6
職業	医療
HP	2.85
攻撃力	528
防御力	221
術耐性	0
再配置	2.85
コスト	2.85
ブロック数	2.85
攻撃速度	2.85
素質1	自己修復：攻撃範囲内の地面マスに自身のみが治療可能な「再構成核」を1体配置可能。 再構成核の周囲にいる味方ユニットの攻撃力+15%(+5%)、再構成核が自身かMon3trによる治療を受けると、減衰と跳躍回数の消費なしで次の対象へと跳躍
素質2	戦術連携：自身または再構成核による治療時、自身と治療対象の攻撃速度が10秒間+20(+2)（重複不可）
スキル1	過圧力連結：次に味方を治療する時、対象のHPを自身の攻撃力の200%回復し、その治療の跳躍回数+1
スキル2	超飽和治療：再構成核を優先して治療する。再構成核はMon3trの治療を受けるたび、跳躍治療を追加で1回行う。第二素質の効果が2.8倍にまで上昇する
スキル3	メルトダウン：攻撃範囲変化、再構成核の位置に移動し、攻撃力+330%、攻撃間隔が短縮(-1.5)、ブロック数+2、最大HP+5000。1秒ごとにHPを80失い、 ブロック中の敵全員を同時に攻撃。通常攻撃が対象に確定ダメージを与えるようになり、攻撃時に自身のHPを攻撃力の50%治療する スキル終了時または致命的なダメージを受けた際、元の位置に戻る

一覧に戻る
削除する
編集する

図 5.4 詳細表示ページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title><%= data[propertySettings.main] %></title>
6   </head>
7   <body>
8     <h2><%= data[propertySettings.main] %>の詳細</h2>
9     <table border="1">
10      <tr><th>項目</th><th>データ</th></tr>
11      <tr><td>ID</td><td><%= data.id %></td></tr>
12      <% for (const property of properties ) { %>
```

```
13          <% if (property.type === 'url') { %>
14            <tr><td><%= property.label %></td><td><a href="<%= data[property.
                  propertyName] %>"><%= data[property.propertyName] %></a></td><tr
                  >
15          <% } else if (property.type === 'textarea') { %>
16            <tr><td><%= property.label %></td><td><pre><%= data[property.
                  propertyName] %></pre></td></tr>
17          <% } else { %>
18            <tr><td><%= property.label %></td><td><%= data[property.propertyName
                  ] %></td></tr>
19          <% }; %>
20        <% }; %>
21      </table>
22      <a href="/db">一覧に戻る</a><br>
23      <a href="/db_remove/<%= data.id%>">削除する</a><br>
24      <a href="/db_edit/<%= data.id%>">編集する</a><br>
25    </body>
26    </html>
```

2.3.1 タイトル変更機能

5 行目ではタイトルを，8 行目では見出しを propertySettings.main で指定された項目の値を表示するようにしている．

2.3.2 ページ遷移機能

- 一覧表示ページへの遷移 (22 行目):「一覧に戻る」リンクをクリックすると，一覧表示ページ (db.ejs) へ移動する．これは app.js の'/db' に対応する．
- 削除ページへの遷移 (23 行目):「削除する」リンクをクリックすると，削除ページ (db_remove.ejs) へ移動する．これは app.js の'/db_remove' に対応する．
- 編集ページへの遷移 (24 行目):「編集する」リンクをクリックすると，編集ページ (db_edit.ejs) へ移動する．これは app.js の'/db_edit' に対応する．

2.3.3 詳細表示機能

9 行目から table を作成し，詳細表示している．id は必ず表示する項目として 11 行目に直接記述し，その他の項目は 12 行目より properties をもとに表示している．
各情報は以下の優先順位で表示形式が決定される．

- url: タイプが url の場合は実際に url にアクセスできるようになっている．
- textare: タイプが textarea の場合は情報が¡pre¿タグで囲まれて表示されるようになっている．

- その他: 情報がそのまま表示される.

これらの表示形式は共存しない.

2.4 db_add.ejs

2.4.1 db_add.ejs の概要と全体

db_add.ejs は新たなキャラを追加するためのページであり，キャラの各項目ごとの情報を入力できる機能を持つ.

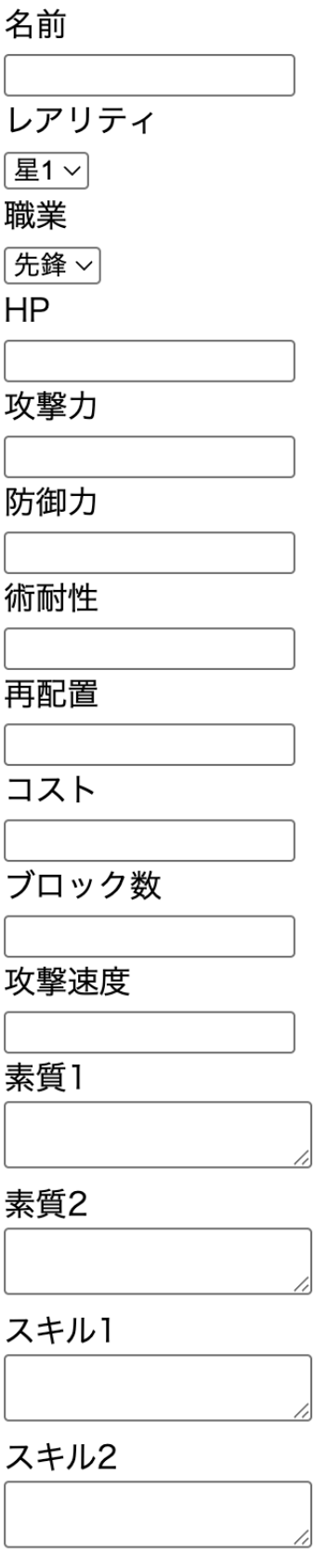

名前
レアリティ
星1
職業
先鋒
HP
攻撃力
防御力
術耐性
再配置
コスト
ブロック数
攻撃速度
素質1
素質2
スキル1
スキル2

図 5.5 追加ページ

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>追加</title>
</head>
<body>
  <form action="/db_add_complete">
    <% for (const property of properties) { %>
      <% if (property.type === 'text') { %>
        <label for="<%= property.propertyName %>"><%= property.label%></label><br>
        <input type="<%= property.type %>" name="<%= property.propertyName %>" id="<%= property.propertyName %>"
        <% if (property.required !== undefined && property.required === true) { %>required<% }; %>
        ><br>
      <% }; %>

      <% if (property.type === 'number') { %>
        <label for="<%= property.propertyName %>"><%= property.label %></label><br>
        <input
          type="number"
          name="<%= property.propertyName %>"
          id="<%= property.propertyName %>"
          <% if (property.min !== undefined) { %> min="<%= property.min %>" <% }; %>
          <% if (property.max !== undefined) { %> max="<%= property.max %>" <% }; %>
          <% if (property.step !== undefined) { %> step="<%= property.step %>" <% }; %>
          <% if (property.required !== undefined && property.required === true) { %>required<% }; %>
        ><br>
      <% } %>

      <% if (property.type === 'select') { %>
        <label for="<%= property.propertyName %>"><%= property.label %></label><br>
        <select id="<%= property.propertyName %>" name="<%= property.propertyName %>">
          <% for (const optionData of property.options) { %>
            <option value="<%= optionData %>"><%= optionData %></option>
          <% }; %>
        </select><br>
      <% }; %>

      <% if (property.type === 'url') { %>
        <label for="<%= property.propertyName %>"><%= property.label %></label><br>
        <input type="<%= property.type %>" name="<%= property.propertyName %>" id="<%= property.propertyName %>"
        <% if (property.required !== undefined && property.required === true) { %>required<% }; %>
        ><br>
      <% }; %>

      <% if (property.type === 'time') {%>
```

```
47            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
48            <input type="<%= property.type %>" name="<%= property.propertyName %
                  >" id="<%= property.propertyName %>"
49            <% if (property.min !== undefined) { %> min="<%= property.min %>" <%
                   }; %>
50            <% if (property.max !== undefined) { %> max="<%= property.max %>" <%
                   }; %>
51            <% if (property.step !== undefined) { %> step="<%= property.step %>"
                   <% }; %>
52            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
53            ><br>
54          <% }; %>
55
56          <% if (property.type === 'textarea') { %>
57            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
58            <textarea name="<%= property.propertyName %>" id="<%= property.
                  propertyName %>"
59            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
60            ></textarea><br>
61          <% }; %>
62        <% }; %>
63        <input type="submit">
64      </form>
65    </body>
66    </html>
```

2.4.2 情報入力機能

8 行目よりフォームを作成し，オブジェクトのタイプごとに入力欄を生成している．タイプごとに生成される入力欄は以下のとおりである．

- 'text'(10 行目): テキスト入力用の`<input type="text">`を生成する．
- 'number'(17 行目): 数値入力用の`<input type="number">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'select'(30 行目): 選択式入力のための`<select>`を生成する．選択肢は options より参照する．
- 'url'(39 行目): URL 入力用の`<input type="url">`を生成する．
- 'time'(46 行目): 時刻入力用の`<input type="time">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'textarea'(56 行目): 複数行テキスト入力用の`<textarea>`を生成する．

'select' タイプを除く各項目において required が true であれば required 属性を付与する．

2.4.3 情報送信機能

63行目では送信ボタンとして`<input type="submit">`を生成し，app.jsの'/db_add_complete'に入力した情報を送信する．

2.5 db_edit.ejs

2.5.1 db_edit.ejsの概要と全体

db_edit.ejsはキャラの情報を編集するためのページであり，キャラの各項目ごとの情報を入力できる機能を持つ．

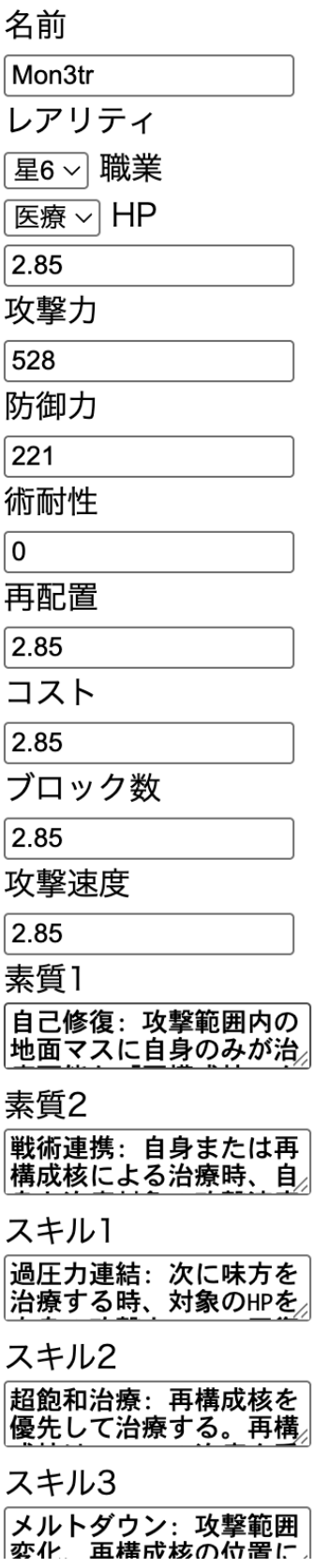


図 5.6 編集ページ

```
1    <!DOCTYPE html>
```

```
2     <html lang="ja">
3     <head>
4       <meta charset="UTF-8">
5       <title>編集</title>
6     </head>
7     <body>
8       <form action="/db_edit_complete/<%= data.id%>">
9         <% for (const property of properties) { %>
10          <% if (property.type === 'text') { %>
11            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
12            <input type="<%= property.type %>" name="<%= property.propertyName %
                  >" id="<%= property.propertyName %>" value="<%= data[property.
                  propertyName]%>"
13            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
14            ><br>
15          <% }; %>
16
17          <% if (property.type === 'number') { %>
18            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
19            <input
20              type="number"
21              name="<%= property.propertyName %>"
22              id="<%= property.propertyName %>"
23              value="<%= data[property.propertyName]%>"
24              <% if (property.min !== undefined) { %> min="<%= property.min %>"
                    <% }; %>
25              <% if (property.max !== undefined) { %> max="<%= property.max %>"
                    <% }; %>
26              <% if (property.step !== undefined) { %> step="<%= property.step %
                    >" <% }; %>
27              <% if (property.required !== undefined && property.required ===
                    true) { %>required<% }; %>
28            ><br>
29          <% } %>
30
31          <% if (property.type === 'select') { %>
32            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
33            <select id="<%= property.propertyName %>" name="<%= property.
                  propertyName %>">
34              <% for (const optionData of property.options) { %>
35                <option value="<%= optionData %>"
36                  <% if (data[property.propertyName] === optionData) { %>
                        selected<% }; %>
37                ><%= optionData %></option>
38              <% }; %>
39            </select>
40          <% }; %>
41
42          <% if (property.type === 'url') { %>
43            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
44            <input type="<%= property.type %>" name="<%= property.propertyName %
                  >" id="<%= property.propertyName %>" value="<%= data[property.
                  propertyName] %>"
45            <% if (property.required !== undefined && property.required === true
```

```
                ) { %>required<% }; %>
46            ><br>
47          <% }; %>
48
49          <% if (property.type === 'time') {%>
50          <label for="<%= property.propertyName %>"><%= property.label %></label
                ><br>
51          <input type="<%= property.type %>" name="<%= property.propertyName %>"
                 id="<%= property.propertyName %>" value="<%= data[property.
                propertyName] %>"
52          <% if (property.min !== undefined) { %> min="<%= property.min %>" <%
                }; %>
53          <% if (property.max !== undefined) { %> max="<%= property.max %>" <%
                }; %>
54          <% if (property.step !== undefined) { %> step="<%= property.step %>" <
                % }; %>
55          <% if (property.required !== undefined && property.required === true)
                { %>required<% }; %>
56          ><br>
57          <% }; %>
58
59          <% if (property.type === 'textarea') { %>
60            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
61            <textarea name="<%= property.propertyName %>" id="<%= property.
                  propertyName %>"
62            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
63            ><%= data[property.propertyName]%></textarea><br>
64          <% }; %>
65        <% }; %>
66        <input type="submit">
67      </form>
68    </body>
69    </html>
```

2.5.2 情報入力機能

8行目よりフォームを作成し，オブジェクトのタイプごとに入力欄を生成している．タイプごとに生成される入力欄は以下のとおりである．

- 'text'(10行目): テキスト入力用の`<input type="text">`を生成する．
- 'number'(17行目): 数値入力用の`<input type="number">`を生成する．min，max，stepが設定されている場合は属性として付与する．
- 'select'(31行目): 選択式入力のための`<select>`を生成する．選択肢はoptionsより参照する．
- 'url'(42行目): URL入力用の`<input type="url">`を生成する．
- 'time'(49行目): 時刻入力用の`<input type="time">`を生成する．min，max，stepが設定されている場合は属性として付与する．

- 'textarea'(59 行目): 複数行テキスト入力用の<textarea>を生成する.

'select' タイプを除く各項目において required が true であれば required 属性を付与する. さらに, 編集前のオブジェクトの値が value="..."や<textarea>...</textarea>として各入力欄の初期値に設定される.

2.5.3 情報送信機能

66 行目では送信ボタンとして<input type="submit">を生成し, app.js の'/db_edit_complete/:id' に入力した情報を送信する.

2.6 db_remove.ejs

2.6.1 db_remove.ejs の全体と概要

db_remove.ejs はキャラを削除する際の最終確認をするページであり, キャラを削除するか選択できる機能を持つ.

図 5.7 削除ページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>削除</title>
6   </head>
7   <body>
8     <h2>本当に「<%= data[propertySettings.main]%>」を削除しますか？ </h2>
9     <p><a href="/db/<%= data.id%>">キャンセル</a></p>
10    <p><a href="/db_remove_complete/<%= data.id%>">削除する</a></p>
11  </body>
12  </html>
```

2.6.2 オブジェクト名表示機能

8 行目では, プログラム上ではキャラをオブジェクトとして認識しているので, ユーザーが理解できるように propertySettings.main で指定された項目の値を表示するようにして

いる．

2.6.3 ページ遷移機能

- 詳細表示ページへの遷移 (9 行目):「キャンセル」リンクをクリックすると，詳細表示ページ (db_detail.ejs) へ移動する．これは app.js の’/db/:id’ に対応する．
- 削除処理の実行 (10 行目):「削除する」リンクをクリックすると，削除処理が実行され，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db_remove_complete’ に対応する．

2.7 db_error.html

2.7.1 db_error.html の概要と全体

db_error.ejs はオブジェクトが取得できなかった時に表示されるページであり，エラー動作から通常の動作へ戻す機能を持つ．

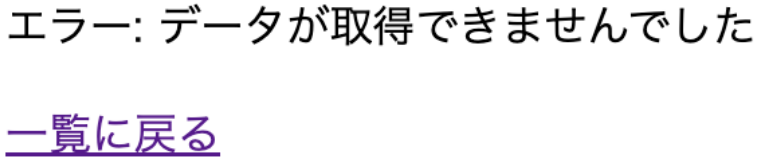


図 5.8 エラーページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>エラー</title>
6   </head>
7   <body>
8     <p>エラー: データが取得できませんでした</p>
9     <p><a href="/db">一覧に戻る</a></p>
10  </body>
11  </html>
```

2.7.2 ページ遷移機能

一覧表示ページへの遷移 (9 行目): 「一覧に戻る」リンクをクリックすると，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db’ に対応する．

6. Overwatchキャラ一覧 開発者向け仕様書

1 サービス概要

Overwatch キャラ一覧とは，FPS ゲーム，Overwatch2 に登場する多くのキャラの情報をまとめたサービスである．

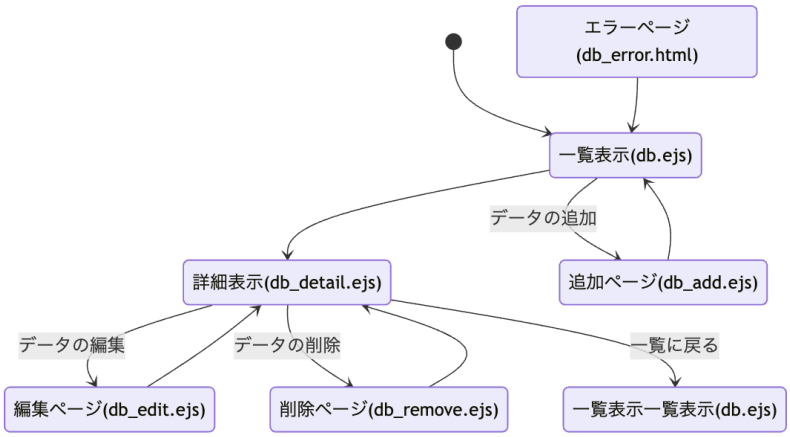


図 6.1 ページ遷移図

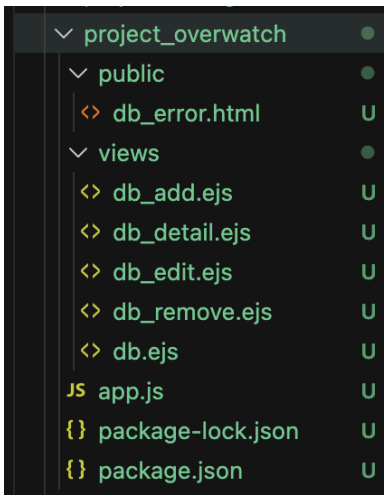


図 6.2 フォルダ構成

2 プログラムの説明

2.1 app.js

2.1.1 app.js の概要と全体

app.js は，データの管理やページ遷移など，このサービスにおけるメインの処理を務める．

```
1   "use strict";
2   const strict = require('assert/strict');
3   const express = require('express');
4   const app = express();
5
6   app.set('view engine', 'ejs');
7   app.use('/public', express.static(__dirname + '/public'));
8
9   let idManager = 0;
10  const properties = [
11    {propertyName: 'name', label: '名前', type: 'text', required: true},
12    {propertyName: 'role', label: 'ロール', type: 'select', options: ['タンク
        ', 'ダメージ', 'サポート']},
13    {propertyName: 'hp', label: 'HP', type: 'number', min: 0, step: 1, view:
        false, required: true},
14    {propertyName: 'mainSkill', label: 'メイン攻撃
        ', type: 'text', view: false},
15    {propertyName: 'subSkill', label: 'サブ攻撃', type: 'text', view: false},
16    {propertyName: 'ability1', label: 'アビリティ
        1', type: 'text', view: false},
17    {propertyName: 'ability2', label: 'アビリティ
        2', type: 'text', view: false},
18    {propertyName: 'ultimate', label: 'アルティメット
        ', type: 'text', view: false},
19    {propertyName: 'passive', label: 'パッシブ', type: 'text', view: false},
20  ];
21  const propertySettings = {
22    main: 'name',
23    click: 'name'
24  };
25  let dataList = [];
26
27  setDefaultData();
28
29  app.get('/db', (req, res) => {
30    res.render('db', {dataList, properties, propertySettings} );
31  });
32
33  app.get('/db/:id', (req, res) => {
34    const id = req.params.id;
35    const data = dataList.find(a => a.id == id);
36    if (data === undefined) {
37      error(res);
38      return;
39    };
40    res.render('db_detail', {data, properties, propertySettings} );
41  });
42
```

```
43    app.get('/db_add', (req, res) => {
44      res.render('db_add', {properties});
45    });
46
47    app.get('/db_add_complete', (req, res) => {
48      const data = req.query;
49      addData(data);
50      res.redirect('/db');
51    });
52
53    app.get('/db_remove/:id', (req, res) => {
54      const id = req.params.id;
55      const data = dataList.find(a => a.id == id);
56      if (data === undefined) {
57        error(res);
58        return;
59      };
60      res.render('db_remove',{data, propertySettings});
61    });
62
63    app.get('/db_remove_complete/:id', (req, res) => {
64      const id = req.params.id;
65      removeData(id);
66      res.redirect('/db');
67    });
68
69    app.get('/db_edit/:id', (req, res) => {
70      const id = req.params.id;
71      const data = dataList.find(a => a.id == id);
72      if (data === undefined) {
73        error(res);
74        return;
75      };
76      res.render('db_edit', {data, properties});
77    });
78
79    app.get('/db_edit_complete/:id', (req, res) => {
80      const id = req.params.id;
81      const data = req.query;
82      const edit = editData(id, data);
83      if (edit === false) {
84        error(res);
85        return;
86      }
87      res.redirect(`/db/${id}`);
88    });
89
90    app.listen(8080, () => console.log('Example app listening on port 8080!'));
91
92    function getId() {
93      idManager++;
94      return idManager;
95    };
96
97    function addData(data) {
98      data['id'] = getId();
99      dataList.push(data);
100   };
101
```

```
102     function removeData(id) {
103       dataList = dataList.filter(a => a.id != id);
104     };
105
106     function editData(id, data) {
107       const dataIndex = dataList.findIndex(a => a.id == id);
108       if (dataIndex === -1) return false;
109       for (const {propertyName} of properties) {
110         dataList[dataIndex][propertyName] = data[propertyName];
111       };
112       return true;
113     };
114
115     function error(res) {
116       res.redirect('/public/db_error.html');
117     };
118
119     function setDefaultData() {
120       const defaultData = [
121         ['キャラ名', 'ロール', 'HP', 'メイン攻撃', 'サブ攻撃', 'アビリティ 1', 'アビ
              リティ 2', 'アルティメット', 'パッシブ'],
122         ['キャラ名', 'ロール', 'HP', 'メイン攻撃', 'サブ攻撃', 'アビリティ 1', 'アビ
              リティ 2', 'アルティメット', 'パッシブ'],
123         ['キャラ名', 'ロール', 'HP', 'メイン攻撃', 'サブ攻撃', 'アビリティ 1', 'アビ
              リティ 2', 'アルティメット', 'パッシブ'],
124         ...
125
126       ];
127       for (const data of defaultData) {
128         const fixedData = {};
129         for (let i = 0; i <properties.length; i++) {
130           fixedData[properties[i].propertyName] = data[i];
131         };
132         addData(fixedData);
133       };
134     };
```

2.1.2 使用する変数

- idManger(9 行目): 各キャラのオブジェクトごとに割り当てる固有の id を管理する変数.
- properties(10 行目): キャラが持つ複数の情報を各項目ごとに管理，設定する定数.

表 6.1 properties 内のオブジェクトで使用可能な要素一覧

要素名	役割	必須/任意
propertyName	プログラム内での項目名を設定する.	必須
label	ブラウザ上で表示する項目名を設定する.	必須
type	項目の情報種別を設定する．また，これによってキャラの追加や編集時の情報の入力方法が変化する．使用可能なタイプは表 6.2 に示す.	必須
required	その項目が入力必須項目かを設定する. undefined，または false で任意．true で必須.	任意
view	一覧表示ページで項目を表示するか設定する. undefined，または true で表示．false で非表示.	任意

表 6.2 type で使用可能なタイプ一覧

タイプ	役割	各タイプが持つ固有要素		
		要素名	役割	任意/必須
'text'	1 行のみの文字列の情報を扱う.	-	-	-
'number'	数値の情報を扱う.	min	数値のみ入力可能. 指定可能な最小値を設定する.	任意
		max	数値のみ入力可能. 指定可能な最大値を設定する.	任意
		step	数値のみ入力可能. 指定可能な数値の間隔を設定する.	任意
'select'	ドロップダウンで特定の選択肢内の情報を扱う.	options	配列のみ入力可能. 選択肢を設定する.	必須
'url'	URL の情報を扱う.	-	-	-
'time'	時間の情報を扱う.	min	時刻文字列('00:00:00') のみ入力可能. 指定可能な最小時間を設定する.	任意
		max	時刻文字列('00:00:00') のみ入力可能. 指定可能な最大時間を設定する.	任意
		step	数値のみ入力可能. 指定可能な時間(秒)の間隔を設定する.	任意
'textarea'	改行ありの文字列の情報を扱う.	-	-	-

- propertySettings(21 行目): サービス全体において，特定の役割を果たす項目を設定する定数.

表 6.3 propertySettings で使用可能な要素一覧

要素名	役割
main	項目名のみ入力可能. キャラのオブジェクトをブラウザ上でユーザーに向けて表す時，代表となる項目を設定する.
click	項目名のみ入力可能. 一覧表示ページから詳細表示ページに移動するためのクリック機能をどの項目に割り当てるのかを設定する.

- dataList: キャラのオブジェクトを保存しておくための変数.

2.1.3 ページ遷移機能

- '/db'(29 行目): 一覧表示ページ (db.ejs) を呼び出す機能を持つ.
  一覧表示ページ (db.ejs) に dataList，properties，propertysettings を渡して呼び出す.

- '/db/:id'(33 行目): 詳細表示ページ (db_detail.ejs) を呼び出す機能を持つ.
  url の id からキャラのオブジェクトを定数 dta として取得し, 詳細表示ページ (db_detail.ejs) に data，properties，propertySettings を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_add'(43 行目): 追加ページ (db_add.ejs) を呼び出す機能を持つ.
  追加ページ (db_add.ejs) に properties を渡して呼び出す.

- '/db_add_complete'(47 行目): キャラの追加処理を行う機能を持つ.
  追加ページ (db_add.ejs) から入力された情報を定数 data として受け取り，関数 addData に data を渡して追加処理をする.
  終了後は'/db' にリダイレクトする.

- '/db_remove/:id'(53 行目): 削除ページ (db_remove.ejs) を呼び出す機能を持つ.
  url の id からキャラのオブジェクトを定数 data として取得し, 削除ページ (db_remove.ejs) に data，propertySettings を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_remove_complete/:id'(63 行目): キャラの削除処理を行う機能を持つ.
  url の id を関数 removeData に渡して削除処理をする.
  終了後は'/db' にリダイレクトする.

- '/db_edit/:id'(69 行目): 編集ページ (db_edit.ejs) を呼び出す機能を持つ.
  url の id からキャラのオブジェクトを定数 data として取得し, 編集ページ (db_edit.ejs) に data，properties を渡して呼び出す.
  id からオブジェクトを取得できなかった場合は関数 error を実行し，return する.

- '/db_edit_complete/:id'(79 行目): キャラの編集処理を行う機能を持つ.
  編集ページ (db_edit.ejs) から入力された情報を定数 data として受け取り，url の id と data を関数 editData に渡して編集処理をする.
  関数 editData から false を受け取った場合は関数 error を実行し，return する.
  終了後は'/db/:id' にリダイレクトする.

2.1.4 使用する関数

- getId(92 行目) 入力: なし，出力: int
  実行されるごとに id として固有の数値を返す機能を持つ.

idManager に 1 を足してその値を返す．

- addData(97 行目) 入力: Object data，出力: なし
  入力された新しいキャラのオブジェクトに id を与えて dataList に保存する機能を持つ．
  関数 getId によって id を与え，dataList に push する．

- removeData(102 行目) 入力: int id，出力: なし
  入力された id をもとに dataList からオブジェクトを削除する機能を持つ．
  入力された id 一致しない id を持つオブジェクトのみで構成される配列を作成し，それを dataList に上書きする．

- editData(106 行目) 入力: int id，Object data，出力: boolean
  dataList 内の既存のオブジェクトの情報を新しいオブジェクトの情報に書き換える機能を持つ．
  入力された id より dataList から既存のオブジェクトの位置を探索し，入力された data より新しいオブジェクトの情報に上書きする．
  dataList 内にオブジェクトが見つからなかった場合は false を，そうでなければ true を返す．

- error(115 行目) 入力: Object res，出力: なし
  エラーページ ('db_error.html') を呼び出す機能を持つ．

- setDefaultData(119 行目) 入力: なし，出力: なし
  dataList に初期値としてオブジェクトを登録する機能を持つ．
  関数内の定数 defaultData は二次元配列になっており，外側の配列内の要素は一つのオブジェクトを表しており，内側の配列内の要素は properties の順番にオブジェクト内の要素として登録されていく．
  定数 fixedData としてオブジェクト化された各要素は fixedData を関数 addData に渡して dataList に追加される．

2.2 db.ejs

2.2.1 db.ejs の概要と全体

db.ejs は複数のキャラ情報の一覧を表示するためのページであり，各キャラの id や設定された項目値を表形式で表示する機能を持つ．また，追加ページや詳細ページへの遷移リンクも提供する．

Overwatchキャラ一覧

キャラを追加する

ID	名前	ロール
1	D.va	タンク
2	ウィンストン	タンク
3	オリーサ	タンク
4	ザリア	タンク
5	シグマ	タンク
6	ジャンカー・クイーン	タンク
7	ドゥームフィスト	タンク
8	ハザード	タンク
9	マウガ	タンク
10	ラインハルト	タンク
11	ラマットラ	タンク
12	レッキング・ボール	タンク
13	ロードホグ	タンク

図 6.3 一覧表示ページ

```
<!Doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>キャラ一覧 Orangestar</title>
</head>
<body>
  <h2>キャラ一覧 Orangestar</h2>
  <p><a href="/db_add">キャラを追加する</a></p>
  <table border="1">
    <tr>
      <th>ID</th>
```

```
13
14          <% for (const property of properties) { %>
15            <% if (property.view === undefined || property.view === true) { %><
                th><%= property.label %></th><% }; %>
16          <% }; %>
17        </tr>
18
19        <% for (const data of dataList) { %>
20          <tbody>
21            <td><%= data.id %></td>
22            <% for (const property of properties) { %>
23              <% if (property.view !== undefined && property.view === false)
                  continue; %>
24              <% if (property.propertyName === propertySettings.click) { %>
25                <td><a href="/db/<%= data.id %>"><%= data[property.propertyName]
                    %></a></td>
26              <% } else if (property.type === 'url') { %>
27                <td><a href="<%= data[property.propertyName] %>"><%= data[
                    property.propertyName] %></a></td>
28              <% } else if (property.type === 'textarea') { %>
29                <td><pre><%= data[property.propertyName] %></pre></td>
30              <% } else { %>
31                <td><%= data[property.propertyName] %></td>
32              <% }; %>
33            <% }; %>
34          </tbody>
35        <% }; %>
36      </table><br>
37    </body>
38    </html>
```

2.2.2 ページ遷移機能

- 追加ページへの遷移 (9 行目):「キャラを追加する」リンクをクリックすると，追加ページ (db_add.ejs) へ移動する．これは app.js の'/db_add' に対応する．
- 詳細表示ページへの遷移: properties のうち，propertySettings.click に設定された項目は一覧表示時にリンクとして表示され，クリックすると詳細ページ (db_detail.ejs) へ移動する．app.js の'/db/:id' に対応する．

2.2.3 一覧表示機能

10 行目から table を作成し，一覧表示している．id は必ず表示する項目として 12 行目に直接記述し，その他の項目は 14 行目より properties の view をもとに表示/非表示を判別している．

19 行目よりオブジェクトの各項目の情報を view をもとに表示/非表示を判別している．各情報は以下の優先順位で表示形式が決定される．

- click: propertySettings の click に指定されている項目は詳細表示ページ (db_detail.ejs) に移動する機能を持ち，クリックすると app.js の'/db/:id' を呼び出す．

- url: タイプが url の場合は実際に url にアクセスできるようになっている.
- textare: タイプが textarea の場合は情報が¡pre¿タグで囲まれて表示されるようになっている.
- その他: 情報がそのまま表示される.

これらの表示形式は共存しない.

2.3 db_detail.ejs

db_detail.ejs は 1 キャラの情報をより詳細に表示するためのページであり，一覧表示ページでの情報に加えて，収録アルバムや，キャラの長さを表形式で表示する機能を持つ.

D.vaの詳細

項目	データ
ID	1
名前	D.va
ロール	タンク
HP	700
メイン攻撃	フュージョン・キャノン: 近距離用のオートマチック兵装。範囲攻撃に適している
サブ攻撃	ディフェンス・マトリックス: 正面からの投射物をブロックする
アビリティ1	ブースター: 進行方向に飛び上がる
アビリティ2	マイクロ・ミサイル: 爆発するロケット弾を一斉に発射する
アルティメット	自爆: メックをオーバーロードさせてから脱出する。メックは一定時間後に爆発する
パッシブ	緊急脱出！: メックが破壊された際、緊急脱出する

一覧に戻る
削除する
編集する

図 6.4 詳細表示ページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title><%= data[propertySettings.main] %></title>
6   </head>
7   <body>
8     <h2><%= data[propertySettings.main] %>の詳細</h2>
9     <table border="1">
10      <tr><th>項目</th><th>データ</th></tr>
11      <tr><td>ID</td><td><%= data.id %></td></tr>
12      <% for (const property of properties ) { %>
13        <% if (property.type === 'url') { %>
14          <tr><td><%= property.label %></td><td><a href="<%= data[property.
              propertyName] %>"><%= data[property.propertyName] %></a></td><tr
              >
15        <% } else if (property.type === 'textarea') { %>
```

```
16            <tr><td><%= property.label %></td><td><pre><%= data[property.
                  propertyName] %></pre></td></tr>
17          <% } else { %>
18            <tr><td><%= property.label %></td><td><%= data[property.propertyName
                  ] %></td></tr>
19          <% }; %>
20        <% }; %>
21      </table>
22      <a href="/db">一覧に戻る</a><br>
23      <a href="/db_remove/<%= data.id%>">削除する</a><br>
24      <a href="/db_edit/<%= data.id%>">編集する</a><br>
25    </body>
26    </html>
```

2.3.1 タイトル変更機能

5 行目ではタイトルを，8 行目では見出しを propertySettings.main で指定された項目の値を表示するようにしている．

2.3.2 ページ遷移機能

- 一覧表示ページへの遷移 (22 行目):「一覧に戻る」リンクをクリックすると，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db’ に対応する．
- 削除ページへの遷移 (23 行目):「削除する」リンクをクリックすると，削除ページ (db_remove.ejs) へ移動する．これは app.js の’/db_remove’ に対応する．
- 編集ページへの遷移 (24 行目):「編集する」リンクをクリックすると，編集ページ (db_edit.ejs) へ移動する．これは app.js の’/db_edit’ に対応する．

2.3.3 詳細表示機能

9 行目から table を作成し，詳細表示している．id は必ず表示する項目として 11 行目に直接記述し，その他の項目は 12 行目より properties をもとに表示している．
各情報は以下の優先順位で表示形式が決定される．

- url: タイプが url の場合は実際に url にアクセスできるようになっている．
- textare: タイプが textarea の場合は情報が¡pre¿タグで囲まれて表示されるようになっている．
- その他: 情報がそのまま表示される．

これらの表示形式は共存しない．

2.4 db_add.ejs

2.4.1 db_add.ejs の概要と全体

db_add.ejs は新たなキャラを追加するためのページであり，キャラの各項目ごとの情報を入力できる機能を持つ．

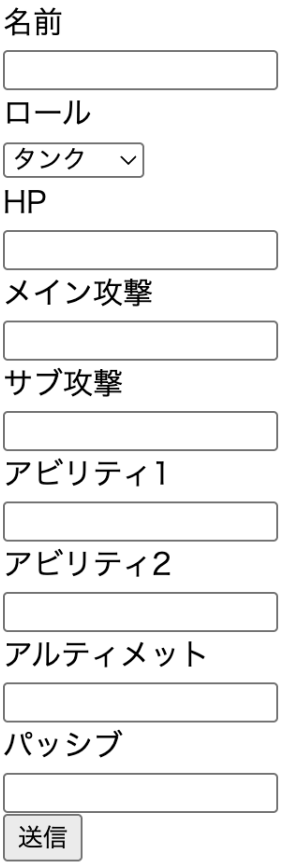


図 6.5 追加ページ

```
1   <!DOCTYPE html>
2   <html lang="ja">
3   <head>
4     <meta charset="UTF-8">
5     <title>追加</title>
6   </head>
7   <body>
8     <form action="/db_add_complete">
9       <% for (const property of properties) { %>
10        <% if (property.type === 'text') { %>
11          <label for="<%= property.propertyName %>"><%= property.label%></
                label><br>
12          <input type="<%= property.type %>" name="<%= property.propertyName %
                >" id="<%= property.propertyName %>"
13          <% if (property.required !== undefined && property.required === true
                ) { %>required<% }; %>
```

```
14            ><br>
15          <% }; %>
16
17          <% if (property.type === 'number') { %>
18            <label for="<%= property.propertyName %>"><%= property.label %></
                label><br>
19            <input
20              type="number"
21              name="<%= property.propertyName %>"
22              id="<%= property.propertyName %>"
23              <% if (property.min !== undefined) { %> min="<%= property.min %>"
                    <% }; %>
24              <% if (property.max !== undefined) { %> max="<%= property.max %>"
                    <% }; %>
25              <% if (property.step !== undefined) { %> step="<%= property.step %
                    >" <% }; %>
26              <% if (property.required !== undefined && property.required ===
                    true) { %>required<% }; %>
27            ><br>
28          <% } %>
29
30          <% if (property.type === 'select') { %>
31            <label for="<%= property.propertyName %>"><%= property.label %></
                label><br>
32            <select id="<%= property.propertyName %>" name="<%= property.
                propertyName %>">
33              <% for (const optionData of property.options) { %>
34                  <option value="<%= optionData %>"><%= optionData %></option>
35              <% }; %>
36            </select><br>
37          <% }; %>
38
39          <% if (property.type === 'url') { %>
40            <label for="<%= property.propertyName %>"><%= property.label %></
                label><br>
41            <input type="<%= property.type %>" name="<%= property.propertyName %
                >" id="<%= property.propertyName %>"
42            <% if (property.required !== undefined && property.required === true
                ) { %>required<% }; %>
43            ><br>
44          <% }; %>
45
46          <% if (property.type === 'time') {%>
47            <label for="<%= property.propertyName %>"><%= property.label %></
                label><br>
48            <input type="<%= property.type %>" name="<%= property.propertyName %
                >" id="<%= property.propertyName %>"
49            <% if (property.min !== undefined) { %> min="<%= property.min %>" <%
                 }; %>
50            <% if (property.max !== undefined) { %> max="<%= property.max %>" <%
                 }; %>
51            <% if (property.step !== undefined) { %> step="<%= property.step %>"
                  <% }; %>
52            <% if (property.required !== undefined && property.required === true
                ) { %>required<% }; %>
53            ><br>
54          <% }; %>
55
56          <% if (property.type === 'textarea') { %>
```

```
57            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
58            <textarea name="<%= property.propertyName %>" id="<%= property.
                  propertyName %>"
59            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
60            ></textarea><br>
61          <% }; %>
62        <% }; %>
63        <input type="submit">
64      </form>
65    </body>
66    </html>
```

2.4.2 情報入力機能

8行目よりフォームを作成し，オブジェクトのタイプごとに入力欄を生成している．タイプごとに生成される入力欄は以下のとおりである．

- 'text'(10行目): テキスト入力用の`<input type="text">`を生成する．
- 'number'(17行目): 数値入力用の`<input type="number">`を生成する．min，max，stepが設定されている場合は属性として付与する．
- 'select'(30行目): 選択式入力のための`<select>`を生成する．選択肢はoptionsより参照する．
- 'url'(39行目): URL入力用の`<input type="url">`を生成する．
- 'time'(46行目): 時刻入力用の`<input type="time">`を生成する．min，max，stepが設定されている場合は属性として付与する．
- 'textarea'(56行目): 複数行テキスト入力用の`<textarea>`を生成する．

'select'タイプを除く各項目においてrequiredがtrueであればrequired属性を付与する．

2.4.3 情報送信機能

63行目では送信ボタンとして`<input type="submit">`を生成し，app.jsの'/db_add_complete'に入力した情報を送信する．

2.5 db_edit.ejs

2.5.1 db_edit.ejsの概要と全体

db_edit.ejsはキャラの情報を編集するためのページであり，キャラの各項目ごとの情報を入力できる機能を持つ．

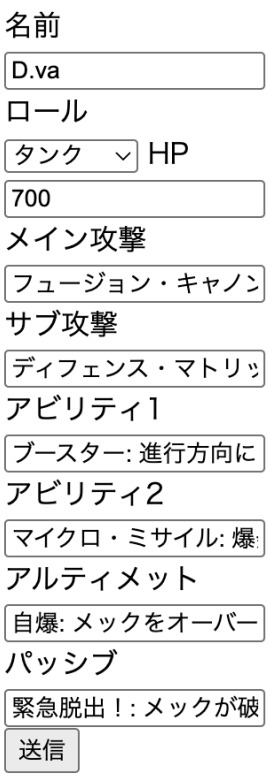


図 6.6 編集ページ

```
1   <!DOCTYPE html>
2   <html lang="ja">
3   <head>
4     <meta charset="UTF-8">
5     <title>編集</title>
6   </head>
7   <body>
8     <form action="/db_edit_complete/<%= data.id%>">
9       <% for (const property of properties) { %>
10        <% if (property.type === 'text') { %>
11          <label for="<%= property.propertyName %>"><%= property.label%></
                label><br>
12          <input type="<%= property.type %>" name="<%= property.propertyName %
                >" id="<%= property.propertyName %>" value="<%= data[property.
                propertyName]%>"
13          <% if (property.required !== undefined && property.required === true
                ) { %>required<% }; %>
14          ><br>
15        <% }; %>
16
17        <% if (property.type === 'number') { %>
18          <label for="<%= property.propertyName %>"><%= property.label %></
                label><br>
19          <input
20            type="number"
21            name="<%= property.propertyName %>"
22            id="<%= property.propertyName %>"
```

```
23              value="<%= data[property.propertyName]%>"
24              <% if (property.min !== undefined) { %> min="<%= property.min %>"
                    <% }; %>
25              <% if (property.max !== undefined) { %> max="<%= property.max %>"
                    <% }; %>
26              <% if (property.step !== undefined) { %> step="<%= property.step %
                    >" <% }; %>
27              <% if (property.required !== undefined && property.required ===
                    true) { %>required<% }; %>
28            ><br>
29          <% } %>
30
31          <% if (property.type === 'select') { %>
32            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
33            <select id="<%= property.propertyName %>" name="<%= property.
                  propertyName %>">
34              <% for (const optionData of property.options) { %>
35                  <option value="<%= optionData %>"
36                    <% if (data[property.propertyName] === optionData) { %>
                          selected<% }; %>
37                  ><%= optionData %></option>
38              <% }; %>
39            </select>
40          <% }; %>
41
42          <% if (property.type === 'url') { %>
43            <label for="<%= property.propertyName %>"><%= property.label %></
                  label><br>
44            <input type="<%= property.type %>" name="<%= property.propertyName %
                  >" id="<%= property.propertyName %>" value="<%= data[property.
                  propertyName] %>"
45            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
46            ><br>
47          <% }; %>
48
49          <% if (property.type === 'time') {%>
50          <label for="<%= property.propertyName %>"><%= property.label %></label
                ><br>
51          <input type="<%= property.type %>" name="<%= property.propertyName %>"
                 id="<%= property.propertyName %>" value="<%= data[property.
                propertyName] %>"
52          <% if (property.min !== undefined) { %> min="<%= property.min %>" <%
                }; %>
53          <% if (property.max !== undefined) { %> max="<%= property.max %>" <%
                }; %>
54          <% if (property.step !== undefined) { %> step="<%= property.step %>" <
                % }; %>
55          <% if (property.required !== undefined && property.required === true)
                { %>required<% }; %>
56          ><br>
57          <% }; %>
58
59          <% if (property.type === 'textarea') { %>
60            <label for="<%= property.propertyName %>"><%= property.label%></
                  label><br>
61            <textarea name="<%= property.propertyName %>" id="<%= property.
                  propertyName %>"
```

```
62            <% if (property.required !== undefined && property.required === true
                  ) { %>required<% }; %>
63            ><%= data[property.propertyName]%></textarea><br>
64          <% }; %>
65        <% }; %>
66        <input type="submit">
67      </form>
68    </body>
69    </html>
```

2.5.2 情報入力機能

8 行目よりフォームを作成し，オブジェクトのタイプごとに入力欄を生成している．タイプごとに生成される入力欄は以下のとおりである．

- 'text'(10 行目): テキスト入力用の`<input type="text">`を生成する．
- 'number'(17 行目): 数値入力用の`<input type="number">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'select'(31 行目): 選択式入力のための`<select>`を生成する．選択肢は options より参照する．
- 'url'(42 行目): URL 入力用の`<input type="url">`を生成する．
- 'time'(49 行目): 時刻入力用の`<input type="time">`を生成する．min，max，step が設定されている場合は属性として付与する．
- 'textarea'(59 行目): 複数行テキスト入力用の`<textarea>`を生成する．

'select' タイプを除く各項目において required が true であれば required 属性を付与する．さらに，編集前のオブジェクトの値が value="..." や`<textarea>...</textarea>`として各入力欄の初期値に設定される．

2.5.3 情報送信機能

66 行目では送信ボタンとして`<input type="submit">`を生成し，app.js の'/db_edit_complete/:id' に入力した情報を送信する．

2.6 db_remove.ejs

2.6.1 db_remove.ejs の全体と概要

db_remove.ejs はキャラを削除する際の最終確認をするページであり，キャラを削除するか選択できる機能を持つ．

図 6.7 削除ページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>削除</title>
6   </head>
7   <body>
8     <h2>本当に「<%= data[propertySettings.main]%>」を削除しますか？ </h2>
9     <p><a href="/db/<%= data.id%>">キャンセル</a></p>
10    <p><a href="/db_remove_complete/<%= data.id%>">削除する</a></p>
11  </body>
12  </html>
```

2.6.2 オブジェクト名表示機能

8 行目では，プログラム上ではキャラをオブジェクトとして認識しているので，ユーザーが理解できるように propertySettings.main で指定された項目の値を表示するようにしている．

2.6.3 ページ遷移機能

- 詳細表示ページへの遷移 (9 行目):「キャンセル」リンクをクリックすると，詳細表示ページ (db_detail.ejs) へ移動する．これは app.js の’/db/:id’ に対応する．
- 削除処理の実行 (10 行目):「削除する」リンクをクリックすると，削除処理が実行され，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db_remove_complete’ に対応する．

2.7 db_error.html

2.7.1 db_error.html の概要と全体

db_error.ejs はオブジェクトが取得できなかった時に表示されるページであり，エラー動作から通常の動作へ戻す機能を持つ．

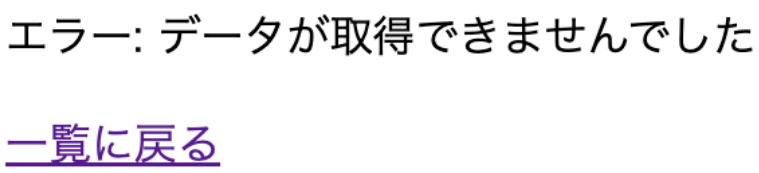


図 6.8 エラーページ

```
1   <!Doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>エラー</title>
6   </head>
7   <body>
8     <p>エラー: データが取得できませんでした</p>
9     <p><a href="/db">一覧に戻る</a></p>
10  </body>
11  </html>
```

2.7.2 ページ遷移機能

一覧表示ページへの遷移 (9 行目): 「一覧に戻る」リンクをクリックすると，一覧表示ページ (db.ejs) へ移動する．これは app.js の’/db’ に対応する．