データベース及び演習 （2023年7月31日）

水谷祐生[1,a)]

概要：本稿は，情報処理学会論文誌ジャーナルに投稿する原稿を執筆する際，および論文採択後に最終原稿を準備する際の注意点等をまとめたものである．大きく分けると，論文投稿の流れと，LaTeX と専用のスタイルファイルを用いた場合の論文フォーマットに関する指針，および論文の内容に関してするべきこと，するべきでないことをまとめたべからずチェックリストからなる．本稿自体も LaTeX と専用のスタイルファイルを用いて執筆されているため，論文執筆の際に参考になれば幸いである．

Database and Exercises (version 2012/10/12)

Yusei Mizutani[1,a)]

1. 機能概要

このアプリケーションには

- 「作成者名」「部屋名」「部屋の説明」を入力した後、「作成」ボタンを押すことで部屋を作成する機能
- 部屋の説明欄を編集する機能
- タグを作成する機能
- 作成したタグを部屋の参加者に割り当てる機能
- 参加者を追加する機能
- 参加者を削除する機能
- 参加者カードの「編集」を押すことで「名前」または「コメント」を編集する機能
- 部屋の参加者の受付をするかしないかを決める機能

といった機能が実装されている．

2. 利用技術

2.1 Next.js

2.1.1 Next.js とは

Next.js とは React をベースに開発された、フロントエンドフレームワークである．

Next.js では画像・レンダリングを最適化する。最適化するメリットとしてページの読み込み速度が高速になることである. また、読み込み速度の高速化は SEO にも効果がある. このように Next.js を利用することで、Web ページ閲覧時のユーザー体験を向上させる、作成したページを誰かに見てもらいやすくなるといったメリットがある．

2.1.2 React とは

React とは Facebook 社が開発した Web サイト上の UI パーツを構築するための JavaScript ライブラリである. また、JSX 記法と呼ばれる構文が採用されていて、これにより一つのファイル内で HTML タグに対する動きを操作することができる. 同様の JavaScript のライブラリとして Vue.js、Angular などが挙げらるが、React は中でも世界的に圧倒的な導入率を誇ってている。

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，情報処理大学
Presently with Johoshori Uniersity
[a)] joho.taro@ipsj.or.jp

2.2 ChakraUI

Chakra UI とは、UI コンポーネントライブラリの 1 つで、自前で CSS を書かなくてもパラメータ指定でスタイルを記述でき、デザインに一貫性を持たせることができる。再利用可能なコンポーネントも多く用意されていて、アラートダイアログやドロワーメニュー、トーストなど自前で作ろうとすると大変なものも簡単に実現できる。また、レスポンシブ対応も容易に行うことができるため誰でも簡単に洗練されたデザインの UI を構築することが可能である.

2.3 TypeScript

TypeScript とはオープンソースかつ、JavaScript を拡張して開発されたプログラミング言語であり、Microsoft によって開発された.

基本的な文法は JavaScript と大した変化はないが、大きな違いとして型付けができるかどうかが挙げられる.

従来までの JavaScript には型付けがない、つまり動的型付けといって実行時にプログラム側が勝手に数値型、文字型を判定するため実行時のエラーの存在に気づかず思わぬバグに繋がることがあった. この問題を解決するため TypeScript では静的型付けといって実行前にプログラマーがあらかじめ型を設定できるようにした. また静的型付けによって、変数の説明ができるため大規模な開発においてソースコードの可読性という観点からも恩恵を受けることができる.

2.4 Go

2.4.1 Go 言語とは

Go 言語とは、シンプルかつ高速な処理が可能なプログラミング言語であり、Google 社によって開発された. コードをシンプルかつにそして高速に動作させるというコンセプトの元に開発されたため、他の言語には実装されている機能、例えば「継承」や「ジェネリクス」などが実装されていない. そのため、誰が書いても似たようなコードになり、アプリケーションの管理を簡単にすることができる.

2.4.2 Gin とは

Gin とは Go 言語の中でも人気のある Web フレームワークである.

Gin には高速なパフォーマンス、JSON のリクエストのバリデーション、ルーティングのグループ化、エラー管理、組み込みのレンダリング、といった特徴があるため、Gin なしで開発するよりもより高速、簡単に Web アプリケーションや API サーバーを開発することができる.

2.4.3 API

API とは「Application Programming Interface」の略である. 言葉通りに意味を解釈すると、アプリケーションをプログラミングするためのインターフェースという意味である. 簡単に説明すると、API とはソフトウェアに API という外部とやりとりする窓口であり、外部アプリとコミュニケーションや連携ができるものである.

3. システム設計

3.1 システム概要

今回のアプリのシステム構成図は図 1 のようになっており、UI 描写専用の Next.js のサーバーと MySQL サーバーとやり取りを行う Go サーバー今回はフレームワークとして Gin を使用しているため Gin サーバーの二つが連携することによって動作するようになっている. 具体的な連携方法として、HTTP メソッドでやり取りをを行っており、今回は Next.js サーバーが GET、POST、PUT、DELETE メソッドにパラメーターやボディーに変数を乗せて Gin サーバーにリクエストを送り、バックエンド側でそれに応じたデータベースに対する操作を行い、最終的な結果を Next.js サーバー、つまりフロントエンドにレスポンスとして返している.

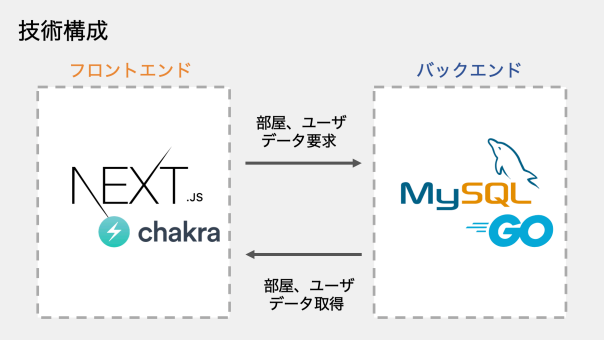

図 1 システム構成図

3.2 技術選定の意図

3.2.1 Next.js

Next.js はデフォルトでサーバーを持っているため、create react app を利用して開発した際に設定しなくてはならない Webpack や babel の記述の負担を減らすことができるため採用した.

3.2.2 ChakraUI

ChakraUI は CSS を記述しなくとも洗練されたデザインの UI コンポーネントを扱うことができるため、フロントエンド側でのデータ処理やクリック時の処理に時間を咲くことができる. すなわち、システム的な機能実装に集中することができるため採用した.

3.2.3 Go

他にも Python、Java などが候補に上がったが、中でも Go 言語は Gorm といった ORM マッパーや、Gin といった強力な Web フレームワークが備わっているため、API サーバーを作成するために便利なパッケージが備わってい

る. それゆえに、Go 言語を採用した

3.2.4 MySQL

今回のアプリの性質上、RDB を用いると最も容易にデータ管理を行うことができると考えたため採用した.

3.3 画面遷移

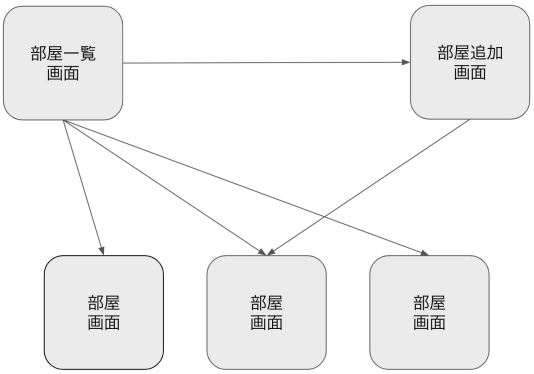

図 2 画面遷移の図

図 2 は今回のプログラムにおける画面遷移を表したものであり、以下にその詳細を記す.

3.3.1 部屋作成の動き

(1) 「部屋一覧画面」で「新しく部屋を作る」ボタンをおして「部屋追加画面」に遷移する
(2) 「部屋追加画面」で作成者名、部屋名、部屋の説明を入力して「作成」ボタンをおす
(3) ボタンを押したと同時に部屋を新規作成、部屋画面に遷移する

といった流れになっている.

3.3.2 部屋参加の動き

(1) 部屋に入っている人から「部屋画面」に表示されている QR コードまたは URL を共有してもらう
(2) 「部屋画面」の「参加者を追加する」ボタンを押すと、ドロワーが表示される
(3) ドロワーにある名前、コメントを入力してから「参加する」ボタンを押す

といった流れになっている.

図 3 は今回のサンプルプログラムにおける管理者が行える操作を表したものであり、以下にその詳細を記す.

(1) 管理者専用の画面の新規登録ボタンを押すと図 2 の新規登録画面に遷移する
(2) 管理者専用画面の中のの会員一覧の更新ボタンを押すと、登録された会員情報の内容を修正することができる更新画面へ遷移する
(3) 管理者専用の画面の中の会員一覧に存在する削除ボタンを押すことで、登録された会員情報を削除することができる、削除確認画面に遷移する

といった流れになっている.

3.4 データベース設計

今回作成したサービスでは三つのテーブルが存在する. 一つ目は、日本の都道府県を閲覧する際に使用される ken テーブル、二つ目は、名前、メールアドレス、パスワードなど会員情報を管理するための member テーブル、三つ目はメールを用いて本登録を行っていないユーザーを仮登録として保持しておくための premember テーブル、これら三つのデータを MariaDB/MySQL を通して行うことでシステムを動かしている.

以下に、それぞれのテーブルについて記述していく.

3.4.1 ken テーブル

```
CREATE TABLE ken (
    id      SMALLINT,
    ken     VARCHAR(10),
    PRIMARY KEY (id)
);
```

上の SQL 文は都道府県のデータを格納するために定義されたテーブルである.

フィールドについて説明すると、id とはカラムを識別するため主キー、ken は都道府県の名前をを格納している.

3.4.2 member テーブル

```
CREATE TABLE member (
    id
    MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
    username    VARCHAR(50),
    password    VARCHAR(128),
    last_name   VARCHAR(50),
    first_name  VARCHAR(50),
    birthday    CHAR(8),
    ken         SMALLINT,
    reg_date    DATETIME,
    cancel      DATETIME,
    PRIMARY KEY (id)
);
```

上の SQL 文は本登録されたユーザーのデータを格納するために定義されたテーブルである.

以下、それぞれのフィールドについて説明する

(1) id とはカラムを識別するため主キー、また UNSIGNED NOT NULLAUTO_INCREMENT とは、カラムを追加される時に自動で順番に id フィールドに数字を格納するという意味である. ただし、0 は受け付けない.
(2) username フィールドは文字列で定義されていて、カッコ内にある 50 とは、50 バイト分の文字列まで格納できるという意味である.
(3) username フィールドは文字列で定義されていて、カッコ内にある 50 とは、50 バイト分の文字列まで格納で

きるという意味である.

(4) password フィールドは文字列で定義されていて、カッコ内にある 128 とは、128 バイト分の文字列まで格納できるという意味である.

(5) last_name フィールドは文字列で定義されていて、カッコ内にある 50 とは、50 バイト分の文字列まで格納できるという意味である.

(6) first_name フィールドは文字列で定義されていて、カッコ内にある 50 とは、50 バイト分の文字列まで格納できるという意味である. また、出身地の情報は都道府県を直接入れているわけではなく、ken テーブルで定義されている id を参照して管理している. こうすることでデータ量の多い都道府県の文字列よりもデータ量の少ない数値で管理できるためサーバーの負荷を軽減することができる.

(7) ken フィールドは数値で定義されていて、SMALLINT とは、-32,767 から 32,767 までの小さい整数を格納することができる意味である.

(8) reg_date フィールドは DARETIME 型で定義されていて、YYYY-MM-DD 形式で格納されている

(9) cancel フィールドは DARETIME 型で定義されていて、YYYY-MM-DD 形式で格納されている

3.4.3 premember テーブル

```
CREATE TABLE premember (
    id
    MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
    username    VARCHAR(50),
    password    VARCHAR(128),
    last_name   VARCHAR(50),
    first_name  VARCHAR(50),
    birthday    CHAR(8),
    ken         SMALLINT,
    link_pass   VARCHAR(128),
    reg_date    DATETIME,
    PRIMARY KEY (id)
);
```

上の SQL 文は仮登録されたユーザーのデータを格納するために定義されたテーブルである.

基本的なフィールドは member テーブルとあまり差異はないが、一部異なる部分もある.link_pass フィールドである. このフィールドは本登録テーブルと仮登録テーブルの情報を結びつけるためのフィールドである.

3.5 システム詳細

表 2 は，type と action の値によって，振り分けられるときの条件を表したものである. 次の画面へ遷移されるときリンクをクリックして画面を表示させるときか、送信ボタンを押して画面を表示させるときのいずれかである. 送信ボタンを押して、画面遷移する場合は、HTTP の GET メソッドにより、パラメーターを URL に載せて、サーバに送信している. 送信ボタンを押して画面遷移する場合は、HTTP の POST メソッドを用いて、ユーザーからは確認できないよう安全に値が送信される.

表 1 各処理における分岐の表

type	action	ボタン	処理	表示画面
無し			認証	会員画面 TOP
無し			未認証	ログイン画面
authenticate	-	-	認証処理 OK	会員画面 TOP
authenticate	-	-	認証処理 NG	ログイン画面
regist	form	-	-	入力画面
regist	confirm	-	入力チェック OK	確認画面
regist	confirm	-	入力チェック NG	入力画面
regist	complete	戻る	-	入力画面
regist	complete	登録	INSERT	完了画面
modify	form	-	-	入力画面
modify	confirm	-	入力チェック OK	確認画面
modify	confirm	-	入力チェック NG	入力画面
modify	complete	戻る	-	入力画面
modify	complete	登録	UPDATE	完了画面
delete	confirm	-	-	入力画面
delete	complete	-	DELETE	完了画面
modify	-	-	ログアウト処理	ログイン画面

ここで、GET メソッドと POST メソッドの違いについて説明する.

GET メソッドは URL にパラメータを含めることで、サーバーに情報を送る HTTP リクエストの一つである. 主に、あるページを取得するためにサーバーにリクエストを送る際にこのメソッドが用いられる.

POST メソッドとはパラメータをリクエストヘッダーに含めることで、サーバーとデータのやり取りを行う HTTP リクエストの一つである.GET メソッドと明確に異なる点として、サーバーに送った情報がユーザーには見ることができない点が挙げられる. この特徴により POST メソッドは、主にデータを追加する際、今回のサンプルプログラムだとユーザーの仮登録、本登録をする際、個人情報などといった第三者に見られてはいけない情報を送信する際に用いられる.

メソッドの決め方としては、$this- type の値でメソッドが決まり、メソッドの内部では$this- action の値で処理を振り分けていく. また，送信ボタンが二つある画面では，ボタンの名称を振り分けに利用している.

3.6 変数設計

3.7 各画面設計

各画面は QuickForm2、Smarty により、PHP ファイルをもとに自動生成されている.

画面描写に必要なファイルは smatry ディレクトリの

templates ディレクトリ内にある.tpl ファイルである. これらのファイルを読み込むことで、画面を描写することができる.

3.7.1 ログイン画面

ログイン画面の描写には、login.tpl を用いている

index.php の情報を参照することでセッション管理を行なっていて、ログイン済みであると認証されなかった場合は、このページに遷移するようになっている.

また、入力欄に入力した内容をブラウザ側で保持しつつ出るために、$form.username.label には, ユーザー名を格納し、$form.username.html はユーザー名の入力欄、HTML のコードに置き換わる. 同様に、$form.password.label はパスワード名に、$form.password.html はパスワー ド入力欄に HTML のコードに置き換わる.

3.7.2 仮登録表示画面

画面の描写には premember.tpl が用いられる.

新規登録画面から確認画面でユーザーが登録情報を確認した後に遷移する画面であり、メールを送信した旨を伝えるための画面である.

3.7.3 会員専用画面

画面の描写には member_top.tpl が用いられる.

ログインを終えた会員と index.php にセッションが残っている場合は子も画面に遷移する.

3.7.4 管理者専用画面

画面の描写には system_login.tpl が用いられる.

管理者用のパスワードを入力し、認証を終えると、このページに遷移する. 会員一覧へのリンクは$SCRIPT_ NAME?type=list& action=form となり$SCRIPT_ NAME は system.php に置き換わる。 クリックすると system.php に対して type=list と action=form が送信され、会員一覧を表示する。

4. 実装

4.1 実装環境

表 2 用いたツール名とそのバージョンの表

名前	バージョン
OS	macOS Venture バージョン 13.4
XAMPP	7.4.28
PHP	7.4.28

今回は以上の表のような環境で動作確認を行なった.

4.2 環境設定

今回のサンプルプログラムを動作させるために必要な XAMPP の環境構築は以下のように行なった.

(1) 公式サイトから XAMPP をインストールする
(2) 設定を行い、XAMPP コントロールパネルを起動させる
(3) Web サーバーである Apache を起動させる

といった流れになっている.

次に実際に今回のシステムを動作させるためのプログラムの配置は以下の手順で行なった

(1) 「環境構築から実践的なシステム作成まで完全習得! PHP7+MariaDB/MySQL マスターブック」に記されているサンプルデータにアクセスする
(2) そこにある Section77 の内容を引用するために、XAMPP ディレ クトリに php_libs を作成し、htdocs と php_libs に引用したサンプルプログラムをすべて移す. データの Section77 に入っているすべてのファイルを移す。
(3) コピペしたフォルダにあるファイルにアクセス権を付与する

MySQL の初期設定は以下の手順で行なった.

(1) sql フォルダを作成する
(2) サンプルプログラムの Section76、Section86 からにある SQL ファイルをコピーして、sql フォルダに格納する
(3) MySQL にアクセスし、予め、データベースやテーブル、初期ユーザーをインサートしておく

といった流れになっている.

今回のサンプルプログラムでは、メールを受け取って、本登録を完了する機能が実装されている. そこで、gmail のメール受け取りに関する設定が必要になったため、その手順を以下に記す.

(1) gmail の Web ページにアクセスする
(2) 右上の歯車のアイコンをクリックする
(3) 押すと、「全ての設定を表示」という項目が表示されるためそれをクリックする
(4) 「メール転送と POP/IMAP」をクリックする
(5) 「IMAP アクセス」のステータスを「IMAP を有効にする」に変更する

以上の手順を踏むと、Gmail のコンソール画面のうち IMAP アクセス」のステータスを「IMAP を有効にする」に変更でき、IMAP サーバーから POP サーバーにメールの内容を送信できるようになる.

また、本来 Smatry を利用する際はインストールするまでに必要な手順を踏む必要があるが、サンプルプログラムを引用することでその必要は無くなったため、説明は省略することにする.

4.3 動作検証

画面遷移の説明の際に用いた図 2 の手順を踏んで行なった. まずは一般会員に関する動作検証を行なった.

(1) ログイン画面では自分の入力した内容がフォームにもそのまま反映されるかを検証した
(2) 新規登録画面ではログイン画面同様、フォームにもそ

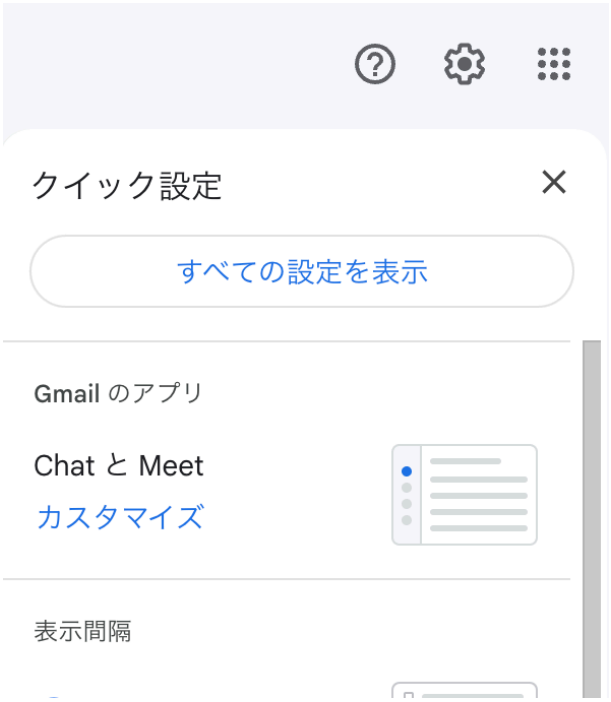

図 3 全ての設定を表示ボタンが表示されている画面の図

図 4 IMAP を有効にした時の画面図

のまま反映されるかに加えデータベースに格納されている都道府県情報が選択できているかを検証した
(3) 登録確認画面ではユーザーが新規登録画面で入力した値が premember テーブルに格納され、表示できているか検証
(4) メール確認後、本登録が完了した旨を伝える画面が表示されるかを検証
(5) 再びログイン画面に遷移した時、新規登録で登録したメールアドレス、パスワードでログインを行えるか検証
(6) 会員専用画面で、登録内容の修正、削除、ログアウトが行えるか検証
(7) 「IMAP アクセス」のステータスを「IMAP を有効にする」に変更する
(8) 「メール転送と POP/IMAP」をクリックする
(9) 「IMAP アクセス」のステータスを「IMAP を有効にする」に変更する

次には管理者に関する動作検証を行なった.
(1) ログイン画面で管理者用のパスワードを入力することで管理者用のページに遷移することができるかを検証した
(2) 管理者ページから会員一覧ページに遷移できるか検証
(3) 会員一覧ページでは、会員情報を全て閲覧できるようになっているか、またそれを削除、修正することができるかを検証
(4) メール確認後、本登録が完了した旨を伝える画面が表示されるかを検証
(5) 再びログイン画面に遷移した時、新規登録で登録したメールアドレス、パスワードでログインを行えるか検証

以上のように一般会員、管理者の動作検証を行ったところ、画面遷移や表示内容、処理について全く不具合が確認できなかったため、システムは正常に動作していると判断した.

5. まとめ

今回のサンプルプログラムでは PHP と MySQL を連携させて作る簡単な会員登録アプリを作成した. 普段、個人またはチームでアプリ開発に取り組む際は、便利なライブラリに頼ったり、Firebase に認証機能を任せていたりしているためメール認証で使われる SMTP サーバー、POP サーバーについて深く知ることができたことはとても良い経験になった.

アプリ全体を通して、いつデータベースにアクセスして削除、追加、閲覧を行ったらよいのか、また、セッションとクッキーの管理、バリテーション、メール送信機能が実装されているため、基本的なアプリの流れを知ることができるアプリであった.

また、今回のアプリは新規登録を行なってログインをして、ログイン者専用ページを表示するといったとてもシンプルなものであったが、今後、会員専用ページに機能を拡張していくことでより面白いアプリにすることができる基盤となるアプリであるとも考えた.