2024 年 12 月 27 日
情報システム工学演習レポート

WebAPI を使用したスマートフォンアプリ

情報経営システム工学分野 B3
学籍番号 ： 24336488
氏名 ： 本間三暉

1　目的

本レポートの目的は，授業の一環として個人で開発したアンケートシステムについて，その企画・設計・実装の全過程を詳細に記録し，そこから得られた技術的知見や自己成長の成果を明確化することである．本システムは，ユーザーが簡便かつ効率的にアンケートを作成・配布・集計できることを目指し，実用性と操作性を重視して設計された．開発を通じて，Web アプリケーション開発の基礎知識やフレームワークの使用方法だけでなく，データベース設計やセキュリティ対策の重要性についても実践的に学習した．

本レポートでは，これらのプロジェクト全体を振り返り，成功した点や改善すべき点を整理し，次回の開発に生かすための具体的な指針を提示する．特に，データの管理手法，ユーザーエクスペリエンスの向上，および個人開発における効率的な問題解決のアプローチについて考察する．この振り返りを通じて，単なる技術習得にとどまらず，今後のより高度な開発への応用力を高めることを目指す．

2　原理と構成

2.1　Web アプリケーション

図 1 に Web アプリケーションのシステム構成を示す．

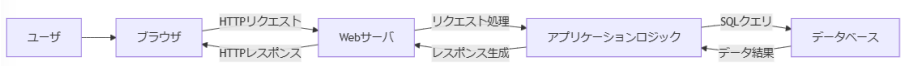


図 1: Web アプリケーションのシステム構成

Web アプリケーションは，クライアント（ユーザが使用するブラウザ）とサーバが通信を行い，データの取得や操作を実現するシステムである．ユーザがブラウザを通じて操作すると，ブラウザは HTTP リクエストをサーバに送信し，サーバはデータベースと連携してリクエストに応じたデータを処理し，HTML や JSON 形式でレスポンスを返す．この一連の流れは以下のように進行する．

2.1.1　クライアントからのリクエスト送信

ユーザがブラウザでボタンをクリックするなどの操作を行うと，JavaScript が動作してサーバにリクエストを送信する．このリクエストは「GET」や「POST」などの HTTP メソッドを使用しており，送信データやクエリが含まれる．

2.1.2　サーバ側での処理

サーバはリクエストを受け取ると，対応するプログラム（アプリケーションロジック）を実行する．ここで，データベースに対して SQL クエリを発行し，必要なデータを取得または更新する．

2.1.3　レスポンスの送信

サーバは処理結果を HTML や JSON 形式に変換し，HTTP レスポンスとしてブラウザに送信する．

2.1.4 ブラウザによる表示

ブラウザは受信したレスポンスを元に画面を更新する．この過程では，JavaScript が動的に表示内容を変更することも可能である．

このように，Web アプリケーションはブラウザを介して動作し，インターネット接続を必要とする．主に HTML や JSON 形式を使用してデータをやり取りするため，デザインや表示の柔軟性に優れている．

2.2 スマートフォンアプリ

図 2 にスマートフォンアプリのシステム構成を示す．

図 2: スマートフォンアプリのシステム構成

スマートフォンアプリは，ネイティブ環境で動作するアプリケーションであり，Web API を利用してサーバとの通信を行う．スマートフォンアプリは，Web アプリケーションと同様にサーバとデータをやり取りするが，データの送受信には主に JSON 形式が用いられる．また，通信の効率化やパフォーマンス向上のために REST API が利用される．

2.2.1 ユーザ操作によるリクエスト送信

ユーザがアプリ内のボタンを押すなどの操作を行うと，アプリが Web API に HTTP リクエストを送信する．このリクエストには，リソースを特定するための情報（例: 商品 ID やユーザデータ）が含まれる．

2.2.2 Web API による処理

API サーバはリクエストを受け取ると，リソースの取得，作成，更新，削除を実行するためにサーバ内のビジネスロジックを実行する．さらに，必要に応じてデータベースと通信してデータを取得または操作する．

2.2.3 レスポンスの受信

API サーバは処理結果を JSON 形式に変換し，スマートフォンアプリに返送する．

2.2.4 アプリ内でのデータ表示

アプリは受信したデータをもとに画面を更新し，ユーザに視覚的に情報を提供する．スマートフォンアプリではネイティブコードを利用するため，リッチなユーザインターフェースを構築できる．

スマートフォンアプリは，Web API を通じた効率的な通信と，オフラインモードなどの高度な機能をサポートしている点が特徴的である．また，データのやり取りには主に JSON 形式が用いられ，軽量かつ扱いやすい．

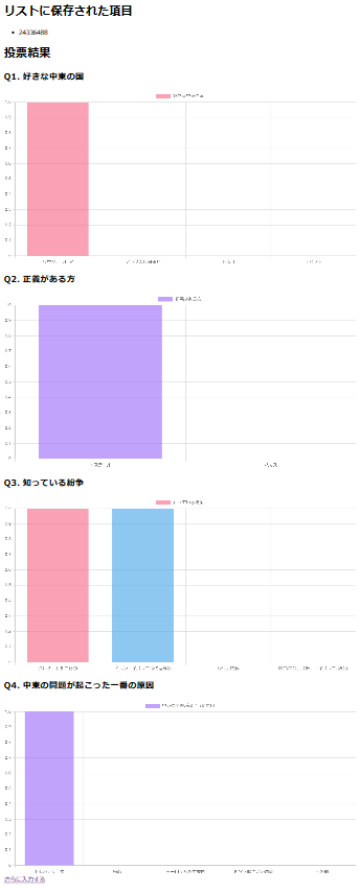
3　動作確認結果

3.1　Web アプリケーション

Web アプリケーションの動作結果を図 3 に示す．

(a) Q1.html

(b) Q1.jsp



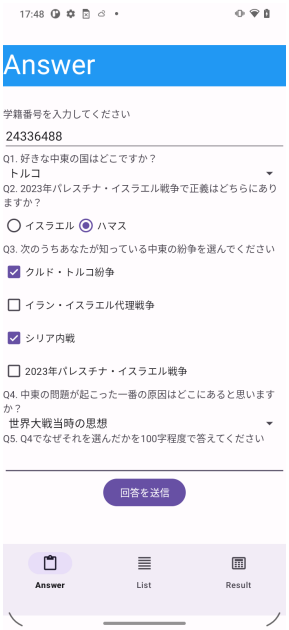
(c) ResultServlet.jsp

図 3: Web アプリケーションの動作結果

しっかり値が入力され，その値を表示できていることがわかる．

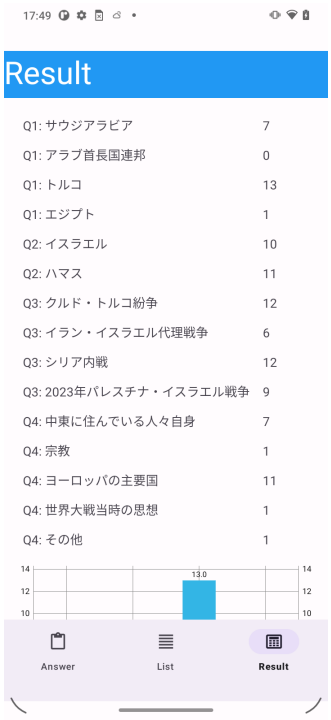
3.2 スマートフォンアプリ

スマートフォンアプリの動作結果を図 4 に示す.

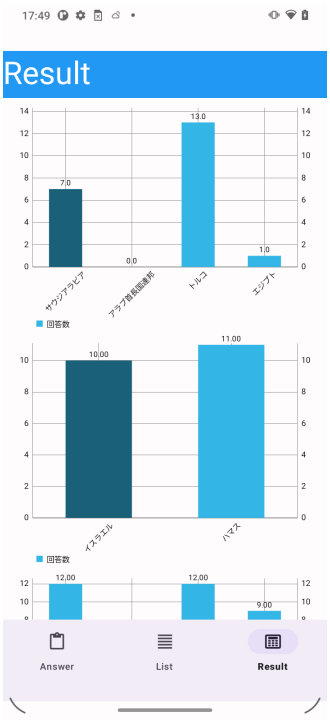


(a) Answer

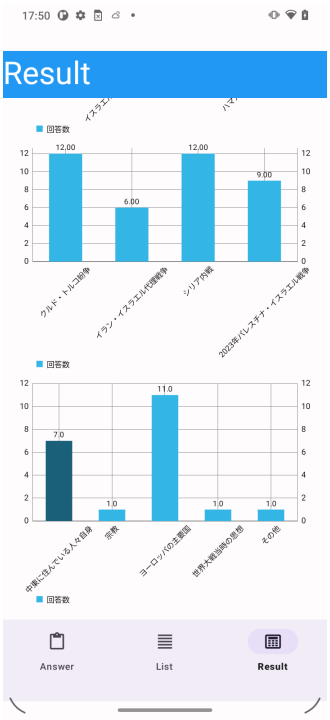
(b) List



(c) Result1



(d) Result2



(e) Result3

図 4: スマートフォンアプリの動作結果

このように，データの入力，リスト表示，グラフ形式での表示に対応しており，動作として十分なものであると言える．

4 考察

4.1 このシステムの長所

このシステムは概ね MVC であると考えられる．MVC のような 3 層アーキテクチャは，アプリケーションをプレゼンテーション層（ユーザインターフェース層），ビジネスロジック層（アプリケーション層），データアクセス層の 3 つに分けて構成する設計手法である．この構成は，それぞれの層に異なる責務を持たせることで，開発や保守の効率性を高めることを目的としている．

この設計の利点の一つは，関心の分離が明確である点である．各層が異なる役割を持つため，コードの理解や修正が容易となり，システム全体の複雑さを軽減できる．また，各層が独立していることで，特定の層を他のプロジェクトや異なる文脈で再利用することも可能であり，再利用性が向上する．

さらに，各層ごとにテストを実施できるため，機能の検証が効率的に行える点も大きな利点である．例えば，ビジネスロジック層のロジックを個別にテストすることで，問題箇所を特定しやすくなる．また，ユーザインターフェースの変更が必要な場合でも，プレゼンテーション層だけを修正すればよいため，他の層への影響を最小限に抑えられる．この柔軟性は，仕様変更や機能追加が頻繁に発生する開発プロジェクトにおいて特に重要である．

加えて，各層が明確に分かれていることで，異なるチームやメンバーが同時並行で作業を進めやすくなり，チーム開発の効率化にも寄与する．このような利点により，3 層アーキテクチャは堅牢で保守性の高いアプリケーション開発に適した構成とされている．

4.2 このシステムの Security

Web サイトを攻撃する方法として以下のものが挙げられる．

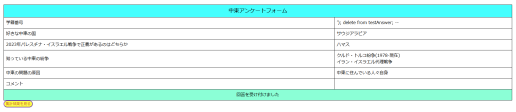
1. SQL インジェクション (SQL Injection)
2. クロスサイトスクリプティング (XSS: Cross-Site Scripting)
3. クロスサイトリクエストフォージェリ (CSRF: Cross-Site Request Forgery)
4. DDoS 攻撃 (Distributed Denial of Service)
5. ディレクトリトラバーサル (Directory Traversal)
6. セッションハイジャック (Session Hijacking)
7. ブルートフォース攻撃 (Brute Force Attack)
8. ゼロデイ攻撃 (Zero-Day Attack)
9. フィッシング (Phishing)
10. サーバーサイドリクエストフォージェリ (SSRF: Server-Side Request Forgery)

今回の授業では，システム自体とデータベースを作成することが目的であるので，“SQL インジェクション”，“クロスサイトスクリプティング”，“セッションハイジャック” などについて考察する．

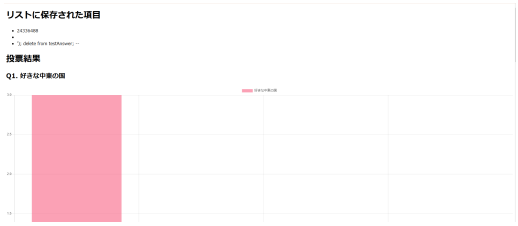
4.2.1 SQL インジェクション

SQL インジェクションとは，悪意のある SQL クエリを入力してデータベースを操作し，データの取得，改ざん，削除を試みる攻撃のことである．今回は**’); delete from testAnswer;** −と**”); delete**

from testAnswer; –というコマンドを入力して攻撃を試みた．今回作った Web ページに攻撃を仕掛けた結果をそれぞれ図 5，6 に示す．



(a) 入力画面



(b) 結果画面

図 5: '); delete from testAnswer; –を入入力した際の様子

(a) 入力画面

(b) 結果画面

図 6: "); delete from testAnswer; –を入力した際の様子

これらの入力を行った後データベースを確認したが，データの欠損などは確認できなかった．また，入力した値がそのまま出力画面に表示されていることが確認できる．

この結果から，変換等が行われ，全く同じ形で送信されていない可能性を考え作成した API から送られてきた JSON ファイルの中身を確認した．JSON ファイルをソースコード 1 に示す．

ソースコード 1: Answer.json

```
1  [
2    {
3      "rid": 1037,
4      "studentId": "24336488",
5      "country0": 1,
6      "country1": 0,
7      "country2": 0,
8      "country3": 0,
9      "isJustice": 0,
10     "know0": 1,
11     "know1": 1,
12     "know2": 0,
13     "know3": 0,
14     "problem0": 1,
15     "problem1": 0,
16     "problem2": 0,
```

```
    "problem3": 0,
    "problem4": 0
  },
  {
    "rid": 1038,
    "studentId": "\u003Cscript\u003Ealert(document.cookie);\u003C/script\u003E",
    "country0": 1,
    "country1": 0,
    "country2": 0,
    "country3": 0,
    "isJustice": 0,
    "know0": 1,
    "know1": 0,
    "know2": 1,
    "know3": 0,
    "problem0": 1,
    "problem1": 0,
    "problem2": 0,
    "problem3": 0,
    "problem4": 0
  },
  {
    "rid": 1039,
    "studentId": "');␣delete␣from␣testAnswer;␣--",
    "country0": 1,
    "country1": 0,
    "country2": 0,
    "country3": 0,
    "isJustice": 1,
    "know0": 1,
    "know1": 1,
    "know2": 0,
    "know3": 0,
    "problem0": 1,
    "problem1": 0,
    "problem2": 0,
    "problem3": 0,
    "problem4": 0
  },
  {
    "rid": 1040,
    "studentId": "\");␣delete␣from␣testItem;␣--",
    "country0": 1,
    "country1": 0,
    "country2": 0,
    "country3": 0,
    "isJustice": 0,
    "know0": 1,
```

```
    "know1": 1,
    "know2": 0,
    "know3": 0,
    "problem0": 1,
    "problem1": 0,
    "problem2": 0,
    "problem3": 0,
    "problem4": 0
  },
  {
    "rid": 1041,
    "studentId": "\"); delete from testAnswer; --",
    "country0": 1,
    "country1": 0,
    "country2": 0,
    "country3": 0,
    "isJustice": 0,
    "know0": 1,
    "know1": 1,
    "know2": 0,
    "know3": 0,
    "problem0": 1,
    "problem1": 0,
    "problem2": 0,
    "problem3": 0,
    "problem4": 0
  },
  {
    "rid": 1042,
    "studentId": "\\\"); delete from testAnswer; --\"",
    "country0": 1,
    "country1": 0,
    "country2": 0,
    "country3": 0,
    "isJustice": 0,
    "know0": 1,
    "know1": 1,
    "know2": 0,
    "know3": 0,
    "problem0": 1,
    "problem1": 0,
    "problem2": 0,
    "problem3": 0,
    "problem4": 0
  },
  {
    "rid": 1043,
    "studentId": "\\\"); delete from testAnswer; --\"",
```

```
    "country0": 1,
    "country1": 0,
    "country2": 0,
    "country3": 0,
    "isJustice": 0,
    "know0": 1,
    "know1": 1,
    "know2": 0,
    "know3": 0,
    "problem0": 1,
    "problem1": 0,
    "problem2": 0,
    "problem3": 0,
    "problem4": 0
  }
]
```

これを見るとわかるように，バックスラッシュやダブルクォーテーションのようなメタ文字がただの文字として出力できるように変更されている．おそらくこれにより不正な SQL 文を送ることができず，正しくデータベースが破壊されなかったと考えられる．

ここで，変換の際にでるバックスラッシュをどうにかしてコマンドではなく文字として出力できれば上手くデータベースを破壊できるのではないかと考え，**\"); delete from testAnswer;** –と入力した．その結果を図 7 に示す．

図 7: \"); delete from testAnswer; –と入力した際の様子

もちろん上手くいかなかった．ソースコード 1 の rid:1042 がこの入力と同じものなのだが，これも図 6 の場合と同様にメタ文字がただの文字として出力できるように変更されているためであると考えられる．

4.2.2 クロスサイトスクリプティング

クロスサイトスクリプティングとは，ユーザー入力に悪意のあるスクリプトを埋め込み，他のユーザーのブラウザでそのスクリプトを実行させる攻撃のことである．今回は Google Chorome にて**<script>window.location='https://www.yahoo.co.jp/';</script>**と入力して攻撃を試

図 8: 入力した際の結果

みた．入力して送信ボタンを押した際の結果をそれぞれ図 8 に示す．

画像を見れば分かる通り `https://www.yahoo.co.jp/` に遷移した．Web サイトで入力をした際はデータベースに保存される前に遷移したため，ResultServlet.jsp などでこのページに遷移することはなかった．また，図 9 のようにスマートフォンアプリで入力した際はデータ自体がデータベースへ送られることはなかった．

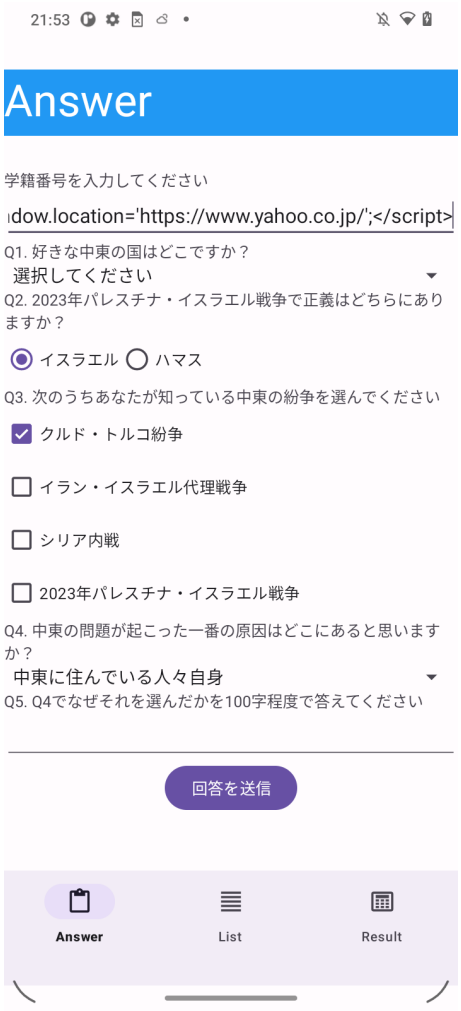


図 9: スマートフォンアプリから XSS を試みた様子

これらのことからわかるように，Web サイトにスクリプトを埋め込むことは可能なので，悪意のあるサイトへの誘導や Web アプリを壊しかねないスクリプトの入力などが考えられる．そのため，メタ文字を入力不可にするなどして対策するべきである．

4.2.3 セッションハイジャック

セッションハイジャックとは，4.2.2 節と同様に，Google Chorome で **<script>alert(document.cookie);</script>** と入力して攻撃を試みた．Q1.jsp や ResultServlet.jsp のような，この文字列が表示され得る場所で出てきたアラートを図 10 に示す．

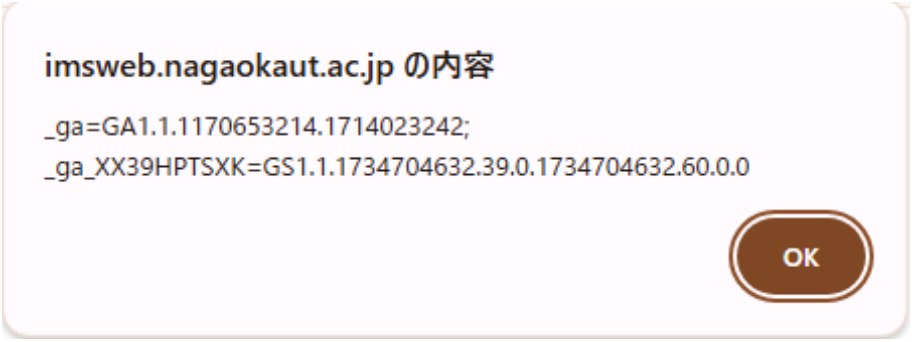


図 10: クロスサイトスクリプティングをした際の出力画面

_ga や_ga_XX39HPTSXK は Google Analytics が利用する Cookie であり，これらの Cookie は匿名化されたデータを使用しているため，個人情報 (名前，住所など) は含まれていない．しかし，ここに表示されている IP アドレスはトラッキングに利用される可能性があるため防ぐことが好ましい．また，このアラートは，out.println() などを用いて表示されるすべての場所で出現したため，4.2.2 節と同様に，悪意のあるサイトへの誘導や Web アプリを壊しかねないスクリプトを入力されてしまう可能性がある．

5 感想

- for QandA と書いてあるのに中身がほぼ ItemManager なのをやめてほしい．そもそも配らないか，しっかり中身を書き換えたものにしてほしい．それか，書き換える部分はスライドに残すなどしてほしい．
-

参考文献

[1] MVC、3 層アーキテクチャから設計を学び始めるための基礎知識 #初心者 - Qiita
https://qiita.com/os1ma/items/7a229585ebdd8b7d86c2