

期末レポート

24G1089 武本龍

2024 年 1 月 7 日

1 はじめに

本報告書は、掲示板システムに「いいね機能」や「投稿時間の自動反映機能」を新たに追加した成果をまとめたものである。利用者、管理者、開発者それぞれの視点に基づき、追加された機能の詳細な説明を行う。さらに、利用者の利便性向上を目的としたユーザーインターフェース（UI）の工夫についても触れ、実際の使用シナリオにおいてどのように役立つかを具体的に示す。本報告書を通じて、掲示板システムの全体像と、新機能がもたらす価値を包括的に理解できるようにすることを目指した。

2 プログラムの概要

今回作成したプログラムの概要を説明する。

2.1 プログラムの保存場所

以下の URL は GitHub のクラウド上に置かれているプログラムである。
https://github.com/Urban-Sea/webpro_06

2.2 プログラムの関係

今回作成したプログラムの関係図を以下に示す。

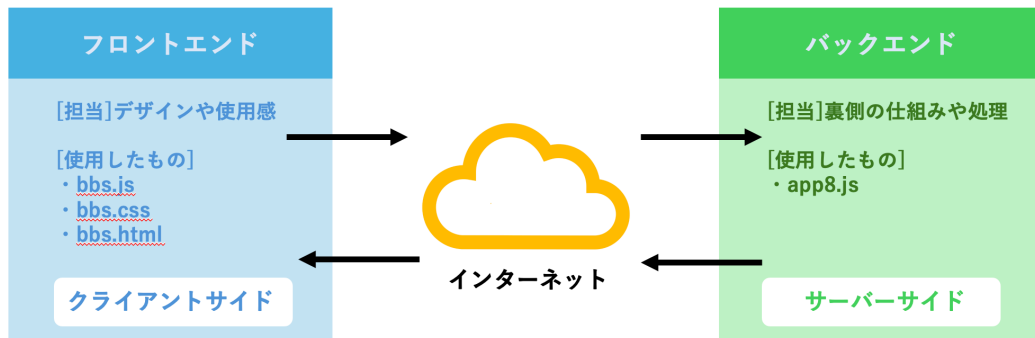


図 1: 今回使用したプログラムの関係

2.3 プログラムの役割

今回作成したプログラムの役割を以下に示す。

2.3.1 app8.js (バックエンド)

- **役割:** サーバーを立ち上げ、リクエストを処理する。
- 投稿の管理 (POST /post), 投稿データの取得 (POST /read), いいね機能 (POST /like) などのエンドポイントを提供。
- データはバックエンド内の bbs 配列に保存される。
- クライアント (フロントエンド) からのリクエストを処理し、レスポンスを返す。

2.3.2 bbs.html (フロントエンドの HTML)

- **役割:** ウェブページの構造を定義する。
- ユーザーが入力するフォームや、投稿一覧の表示部分を構築。
- サーバーと通信するためのスクリプト (bbs.js) を読み込む。
- html だけでは不十分なデザインを取り入れるために css を読み込む。

2.3.3 bbs.html (フロントエンドの CSS)

- **役割:** ウェブページの見た目をデザインする。
- フォーム、投稿のリスト、ボタンなどのスタイル (色, フォント, レイアウト) を指定。

2.3.4 bbs.html (フロントエンドの JavaScript)

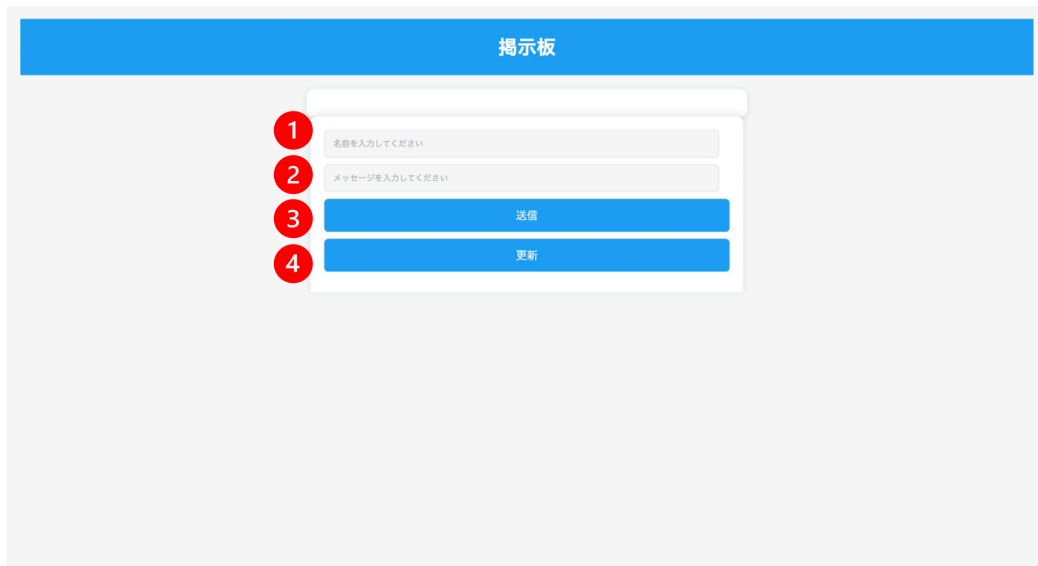
- **役割:** ウェブページの動きを制御する。
- 取得したデータを HTML に動的に反映する (例: 投稿リストの表示, いいね数の更新)。

3 利用者向け

以下は、利用者を対象にした仕様書である。

3.1 掲示板の使い方 1

掲示板の使い方について掲示板の画像をつかって以下で説明する.



The image shows a web interface for a bulletin board. At the top is a blue header bar with the text "掲示板" (Bulletin Board). Below the header is a light gray area containing a white form. The form has four numbered steps indicated by red circles with white numbers: 1. A text input field with the placeholder "名前を入力してください" (Please enter your name). 2. A text input field with the placeholder "メッセージを入力してください" (Please enter your message). 3. A blue button with the text "送信" (Send). 4. A blue button with the text "更新" (Update).

図 2: 使用方法 1

掲示板の基本的な操作手順は、以下の通りである.

1. 名前を入力する.
2. メッセージを入力する.
3. 送信ボタンを押す.
4. 更新ボタンを押す, 内容を反映させる.

また, メッセージを入力しなくても, 他のユーザーが入力したメッセージがあれば, 更新ボタンを押すことでその内容が反映される.

3.2 掲示板の使い方 2

次に掲示板で利用できる機能を掲示板の画像を用いて以下で説明する。



図 3: 使用方法 2

掲示板の基本的な機能は、以下の通りである。

1. 日付：送信された時間を web 上で反映させる
2. いいね：ユーザーがいいねを押すことができる
3. いいねカウント：ユーザーがいいねを押した回数を記録する
4. 投稿件数：すべてのユーザーが投稿した回数を記録する

3.3 掲示板の使い方 3

この掲示板は、パソコン、タブレット、スマートフォンなど、さまざまなデバイスに対応している。



タブレット用の調整768px



スマホ用の調整480px

図 4: 使用方法 3

また、視認性や使いやすさを向上させるためにさらなる機能を追加している。

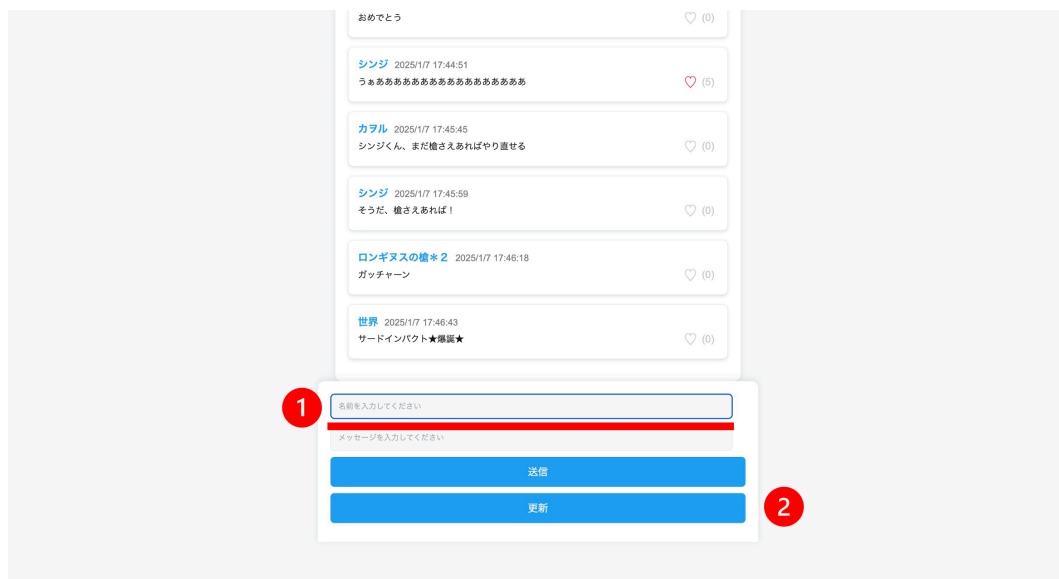


図 5: 使用方法 4

1. フォーム枠 : 選択時に濃い青い枠が表示される
2. 固定スクロール : スクロールしても入力フォームが常に画面下部に固定されている

4 管理者向け

以下は、管理者を対象にした仕様書である。

4.1 サーバーの立ち上げ手順

掲示板で用いるサーバーの立ち上げ方は以下の手順の通りである。

4.1.1 サーバーの立ち上げ

バックエンド側のサーバー（app8.js）を起動する手順は次の通りである。

1. ターミナルを起動し、サーバー処理をするバックエンド側の JavaScript ファイルがあるディレクトリに移動する。（例：cd /Desktop/webpro/webpro_06）
2. 今回は app8.js がサーバー処理をするバックエンド側の JavaScript なので、以下のコマンドを実行する。
node app8.js

4.1.2 telnet によるサーバー接続

別のターミナルを立ち上げて、telnet を使用してサーバーに接続する手順は次の通りである。

1. ターミナルを起動し、以下のコマンドを実行する。
telnet localhost 8080
2. 接続が成功したら、以下のコマンドを順に実行する。
 - (a) HTTP リクエストの開始行を入力する。
GET /bbs HTTP/1.1
 - (b) 次に、ホスト名を指定する。
Host: localhost
3. 最後に、空行（Enter キーを 2 回押す）を送信してリクエストを終了する。

4.2 ログの出力形式及びログの見方

サーバーの動作ログは基本的に以下の通りである。

1. 起動時 :Example app listening on port 8080!
2. 投稿取得時 :GET /BBS
3. 各投稿データ :['name:', 'ゲンドウ', 'message:', 'おい, 課題は終わっているのか?', 'date', '2025/1/7 13:06:38']
4. 読み取られた順序 :read -> X

```
[takemotoryuu@ryu-3:webpro_06]$ node app8.js
Example app listening on port 8080!
GET /BBS
[
  'name:',
  'ゲンドウ',
  'message:',
  'おい, 課題は終わっているのか?',
  'date',
  '2025/1/7 13:06:38'
]
read -> 0
[
  'name:',
  'シンジ',
  'message:',
  '...まだ終わってないよ!',
  'date',
  '2025/1/7 13:07:33'
]
read -> 1
[
  'name:',
  'ミサト',
  'message:',
  'シンジくん, 逃げちゃだめよ!',
  'date',
  '2025/1/7 13:07:57'
]
read -> 2
```

図 6: 管理者

サーバーの動作ログを確認にすることで、投稿者の名前、投稿内容、投稿日時からスパム投稿や不適切な投稿の特定に役立てることができる。

5 開発者向け

以下は、開発者を対象にした仕様書である。以下の3つについて詳しく解説する。

- 「送信」ボタンがクリックされた時の処理 (POST /post)
- 「更新」ボタンがクリックされた時の処理 (POST /check → POST /read)
- 「いいね」ボタンがクリックされた時の処理 (POST /like)
- 「日付」の処理 (POST /post → POST /check → POST /read)
- 「投稿件数」の処理 (POST /post → POST /check)

5.1 「送信」ボタンがクリックされた時の処理

- 役割: 現在の掲示板の投稿数を確認する。
- クライアント側では、投稿数の変化があれば新しい投稿をサーバーから取得し、表示する処理を行う。

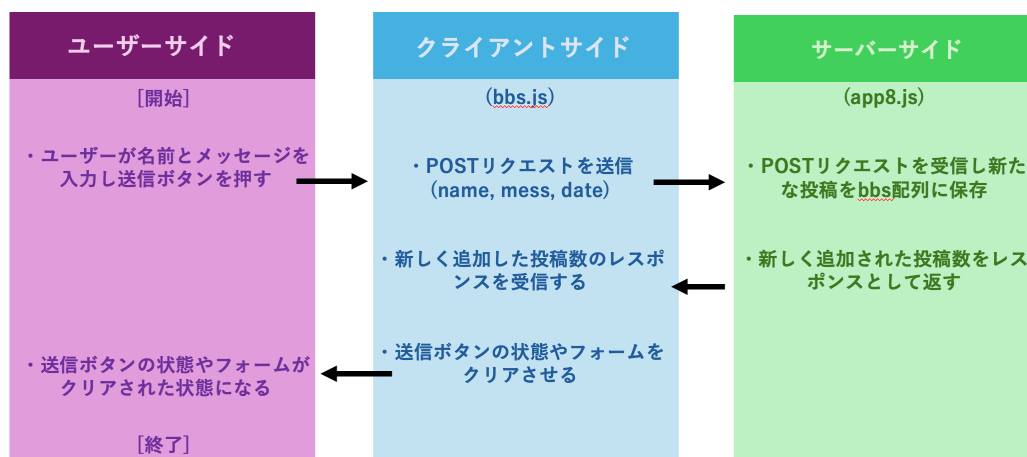


図 7: 送信

5.2 「更新」ボタンがクリックされた時の処理

- 役割: 現在の掲示板の投稿数を確認し、更新する.
- クライアント側では、投稿数の変化があれば新しい投稿をサーバーから取得し、表示する処理を行う.

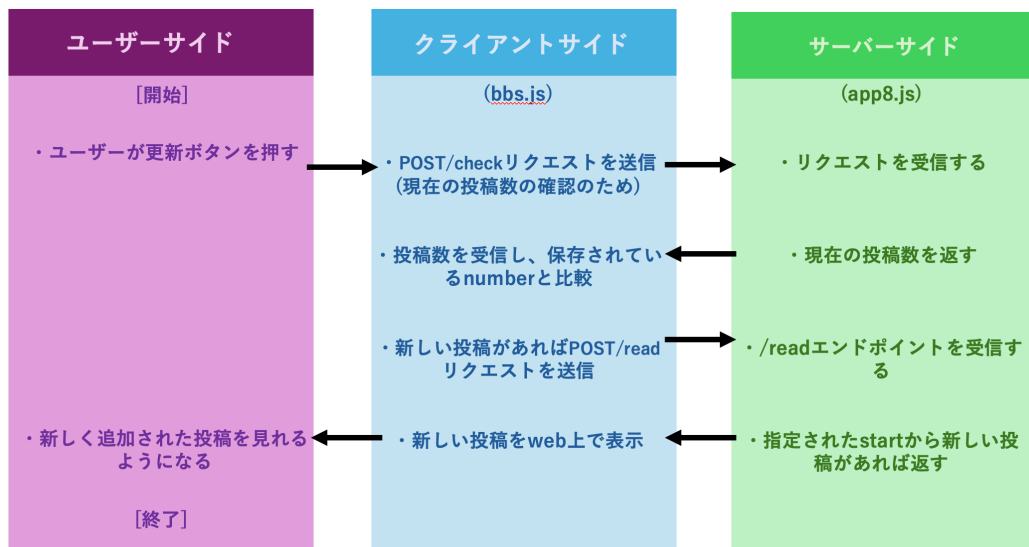


図 8: 更新

5.3 「いいね」 ボタンがクリックされた時の処理

- 役割: 現在の掲示板のいいね数を確認し、一つ一つの投稿に紐づけられた postId を用いて、いいね数を増加させる。
- クライアント側では、ユーザー側に新しい投稿があればサーバーから取得し、表示する処理を行う。
- サーバー側では、bbs 配列内の該当する Id を取得し、いいね数を増加させてレスポンスする。

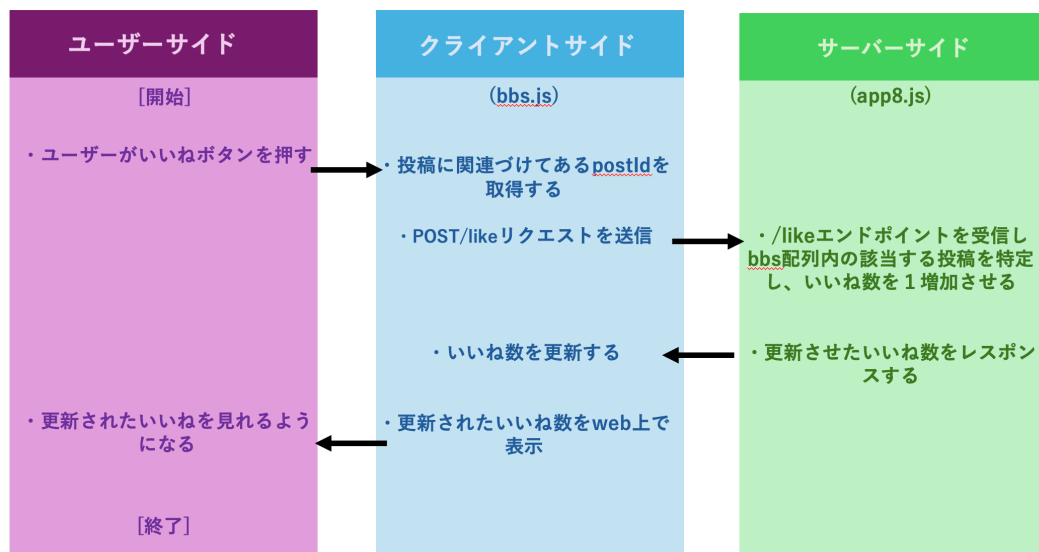


図 9: いいね

5.4 「日付」の処理

app8.js の `app.post("/post")` で使用しているが、JavaScript では、`new Date()` で現在日時を取得でき、その後に `.toLocaleString()` と記入すると、数値を日本のローカルに適した数値表示をすることができる。

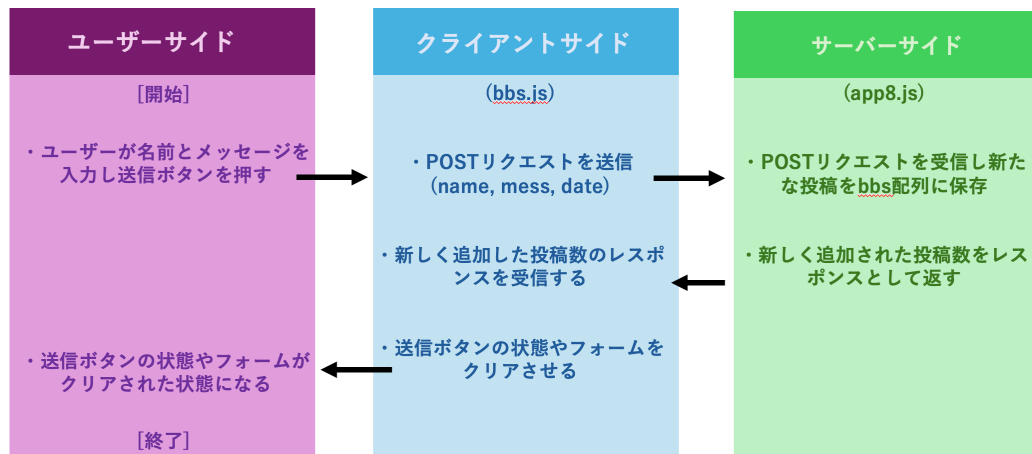


図 10: 日付の送信

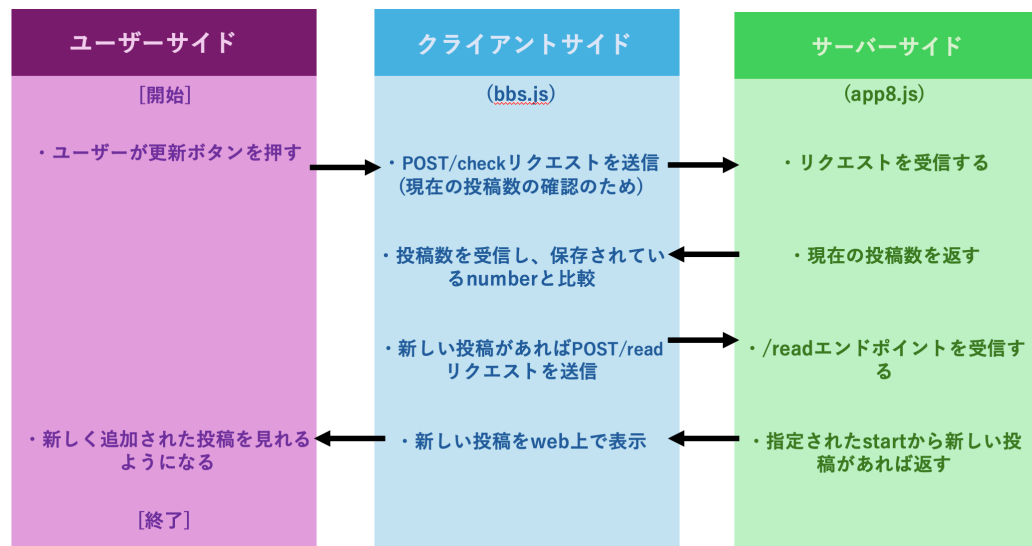


図 11: 日付の更新

日付の更新については、「送信」ボタンがクリックされた時の処理」及び「更新」ボタンがクリックされた時の処理」と全く同じである。

5.5 「投稿件数」の処理

「投稿件数」の処理について以下に示す。

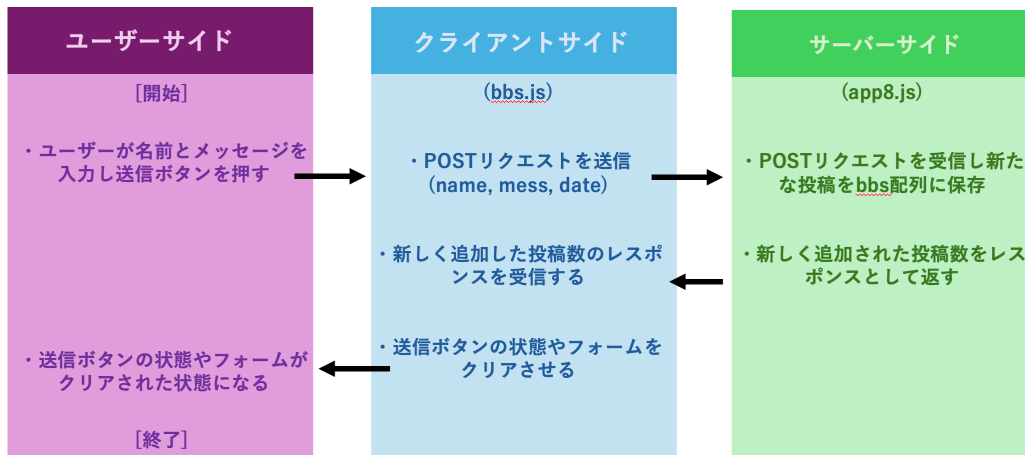


図 12: 送信 (投稿件数)

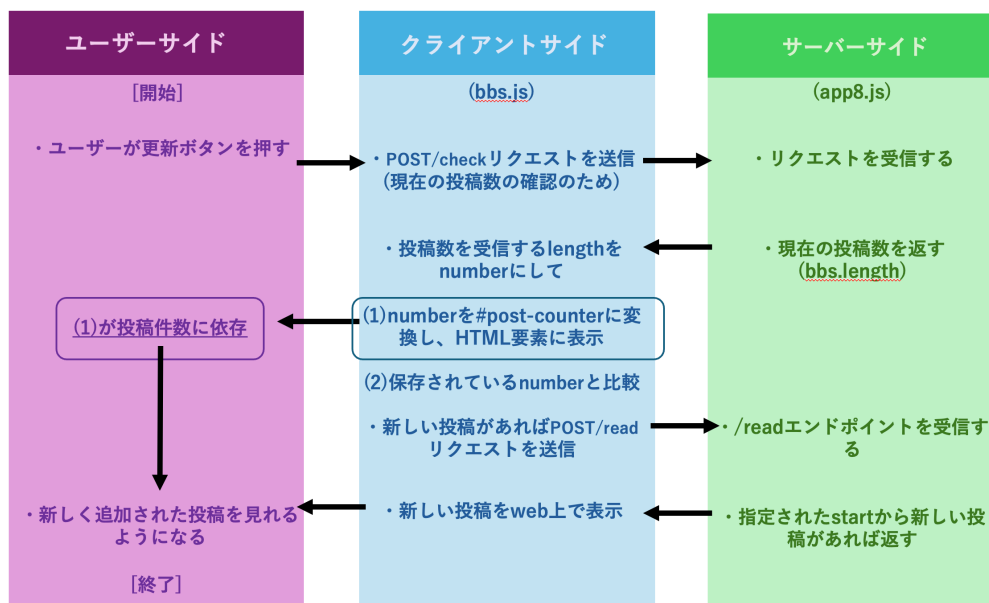


図 13: 更信 (投稿件数)

送信ボタンを押すまでは、「送信」ボタンがクリックされた時の処理」と同じであり、その後の、「更新」ボタンがクリックされた時の処理」は「投稿数を受信する `length` を `number` にして」までは同じであり、その次の (1) だけが HTML 上に表示する投稿件数に依存するため、(2) の処理を行うのではなく、そのまま終了になる。

また、app8.js 内で、元々あった app.post("/check") メソッドを使用して、bbs の配列の長さ (投稿の数) をサーバーからクライアントに送信する。その際、サーバーから帰ってきたデータを value という変数に代入する。その後、

Listing 1: value を post-count に変換

```
1 document.querySelector('#post-count').innerText = 投稿件数: ${value};
```

を用いて、HTML ページ上の post-count 変数に数値を代入し、ページ上に投稿件数がリアルタイムで反映できるようにしている。

6 おわりに

7 参考

7.1 参考サイト

- <https://ics.media/>
- Chat GPT
- YouTube

7.2 参考書籍

- ほんの一手間で劇的に変わる HTML & CSS と Web デザイン実践講座
- HTML5 & CSS3 デザインブック