構造や動作のつながりに着目した Webアプリケーションの可視化

柾山 命[1,a)]　寺田 実[1,b)]

概要：本研究では，フロントエンドが JavaScript，バックエンドが Python で構成されている Web アプリケーションを対象とし，CodeCity という都市メタファーを用いて Web アプリケーションの構造と動作を可視化するシステムを提案する．都市メタファーを用いることで，分散システムである Web アプリケーションの複雑な構造のつながりを理解することが容易になると考える．システムでは、フロントエンドとバックエンド間の通信だけでなく，それぞれのコンポーネント内部での処理も同時に可視化する．クライアントサイドとサーバーサイドのコンポーネントを都市のビルとして，データ通信をコンポーネントである都市のビルとビルの間を行き来するトラックの移動として表す．コンポーネントごとの動作とアプリケーション全体の動作を可視化することで，アプリケーションの構造と一連の処理の理解，エラー原因の特定を容易にすることを目的とする．

キーワード：Web アプリケーション、CodeCity、可視化

1. はじめに

近年，Web アプリケーションの開発数が増加している．Google Chrome のデベロッパーツール [1] など，Web アプリケーション内での動作をコンソールを用いて可視化するものは存在する．しかし，Web アプリケーションの初学者が動作を理解するためには難しい点がある．例えば，Web アプリケーションのクライアントサイドとサーバーサイドのコンポーネント同士のつながりの構造や，クライアントサイドにおける関数の処理がサーバーサイドのどのコンポーネントの関数の処理に作用するかなどの動作のつながりである．この理解の困難さについては 3.2 で記述する．

そこで，Web アプリケーションの初学者が，Web アプリケーションを動かしながら Web アプリケーションの構造や動作のつながりを理解することを支援するシステムを提供し，コンポーネントごとの動作と Web アプリケーション全体の動作を可視化することにより，Web アプリケーションの構造と一連の処理の理解を容易にすることを目的とする．

この目的を達成するため，本研究では，Web アプリケーションの内部構造を都市メタファーを用いて，フロントエンドとバックエンド間で行われるデータ通信だけでなく，それぞれのコンポーネント内部での処理も同時に可視化するシステムを提案する．

[1] 電気通信大学 大学院情報理工学研究科
[a)] h2231115@edu.cc.uec.ac.jp
[b)] terada.minoru@uec.ac.jp

2. 関連研究

2.1 Web アプリケーションの構造の可視化

Cloutier らは，Web アプリケーションのリバースエンジニアリングのために，Web アプリケーションの構造をソースコードの静的な解析とフィルタを用いて可視化する WAVI（WebAppViewer）[2] を提案している．WAVI は HTML のハイパーリンク，JavaScript のクラスやメソッド，関数，変数などの依存関係をソースコードから抽出し，スコープやネストの深さなどでフィルタをかけてテキスト，力学モデルのグラフ，クラス図によって Web アプリケーションの構造を可視化する．

本研究で提案するシステムは都市メタファーを用いることで，クラス間の関係を可視化できるとともに，クラスのコード行数やメソッドの数などの定量的な指標を用いることでより理解を容易にすることが可能であると考える．

2.2 Web アプリケーションで通信されるデータの可視化

画面遷移した際の実行時情報を取得するシステム [3] では，リクエストやメッセージ，データなどを紐付けてログとして抽出できるようにし，システム保守の工程でのエラーの原因調査の工数の削減を行っている．

Charles[4] は Web プロキシを用いたデバッグ支援ツールである．Web アプリケーションを動かしながらクライアントサイドとサーバーサイドで送受信されるデータを記録して表示することができる．

これらの研究では，通信されるデータや呼び出されるクラスやメソッドは文字列でしか表示されない．Web アプリケーションの動作の理解を容易にするためにはグラフィカルな表示が必要であると考える．そこで本研究では，Web アプリケーションのコンポーネント間のデータ通信をトラックの移動というメタファーを用いて表現する．

3. Web アプリケーション

3.1 一般的な Web アプリケーション

Web アプリケーションは，システム利用者の入力に応じてシステム側が処理したデータを利用者に提示するなど，双方向のコミュニケーションを行うことができる Web サイトである．Web アプリケーションはユーザが扱うクライアントサイドと，クライアントからの要求に応じてデータ処理を行うサーバーサイドに分かれる．

3.2 Web アプリケーションの理解に対する困難

Web アプリケーションの構造や動作の理解には以下のような困難がある．

- クライアントサイド，サーバサイドがそれぞれ多数のコンポーネントで構成されているため，構造が不明瞭である．
- 多数のコンポーネントが独立に動くため，クライアントサイドでの動作とサーバーサイドでの動作の連続性を理解することが難しい．例えば，クライアントサイドのどのクラスのメソッドが発火したら，サーバーサイドのどのクラスのメソッドが発火して処理を行っているかが見えないということが挙げられる．
- クライアントサイドとサーバーサイドの間で，具体的にどのようなデータが送受信されているのか簡単に確認することができない．

上記の困難を解決するものとして本研究のシステムを作成する．

3.3 Web アプリケーションフレームワーク

以下に普及している代表的な Web アプリケーションフレームワークを 2 つ紹介する．

3.3.1 Flask

Flask[5] は Python の Web アプリケーションフレームワークである．デフォルトではデータベースの機能がなく，フレームワーク内に搭載されている機能は非常に少ないものであり，学習コストが低い．拡張機能が豊富であるため，フレームワークの仕様に縛られることなく，自身の考えで Web ア

プリケーションを作成していくことが可能である．

3.3.2 Laravel

Laravel[6] は PHP のフレームワークである．Laravel のパッケージを管理する Composer を用いることで，必要となるパッケージを自動でインストールすることが可能である．シンプルなコードで記述できるほか，バリデーションやマイグレーションなどの開発を効率化するための機能も多く備わっている．

Larvel の設計には，MVC モデルという構造が取り入れられている．MVC は

- Model
- View
- Contoroller

の略であり，Model はデータベースとのやり取りを行い，View は画面上への表示を行い，Contoroller は Model と View を制御する橋渡しを行う．機能ごとに分割して作業を行うことができるため，効率的に Web アプリケーションを開発することができる．

4. 都市メタファー

4.1 CodeCity

Wettel ら [7][8][9] は大規模ソフトウェアの分析のために，ソフトウェアの構造を 3D の都市という比喩を用いて視覚化する CodeCity を作成した．CodeCity は Java，C++，Smalltalk を対象の言語としている．図 1 のようにパッケージは都市の地区として表され，入れ子の順序によって彩度が変わる．建物はクラスを表し，高さがメソッド数，底面積が属性数を表し，コードの行数に応じて建物の色が変化する．抽象的なソフトウェア内部の性質を定量化した指標に変換し，内部構造を 1 つの都市として表すことによって，ソフトウェア内部の最もメソッド数が多いクラスの特定における正確性の向上や分析が完了するまでの時間の削減に貢献している．

本研究では，この CodeCity のメタファーの表現を踏襲して Web アプリケーションの構造を可視化する．

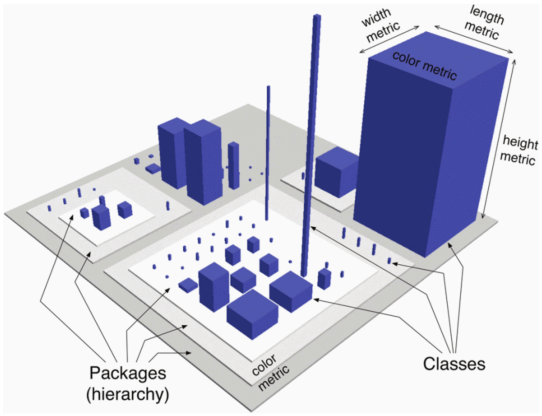

図 1　CodeCity で表されたアプリケーション [8]

4.2 都市メタファーを利用した可視化システム

都市メタファーを用いた可視化システムは CodeCity 以外にも多く存在する．

4.2.1 データとソフトウェアの進化に着目した可視化

Ardigò ら [10] は，都市メタファーを利用してソフトウェア内部の構造に加え，データファイルやデータベースも可視化するシステムである M3triCity を作成している．データベースに存在するデータの参照は，図 2 の A の赤い波線内で青く示されているコードを表す建物と B の赤い波線内のデータベースである赤い円柱を結ぶ，赤と青のグラデーションの線で表される．また，git のリポジトリの URL から時系列順のコミットの情報を得ることができ，図 2 の下部の 6 と注記のあるタイムラインを操作することで，ソフトウェアの進化を辿ることも可能である．

本研究のシステムでは，Web アプリケーションの動作を一時停止する機能を設けることで通信されている JSON データなどの詳細なデータを確認できる機能を設ける．

4.2.2 即時のパフォーマンスの可視化

Heijo[11] は，実行中のアプリケーションのボトルネックやメモリリークなどのパフォーマンスの問題を発見するために都市メタファーを利用している．Heijo は Java アプリケーションや Andoroid アプリケーションを対象としており，パッケージ，

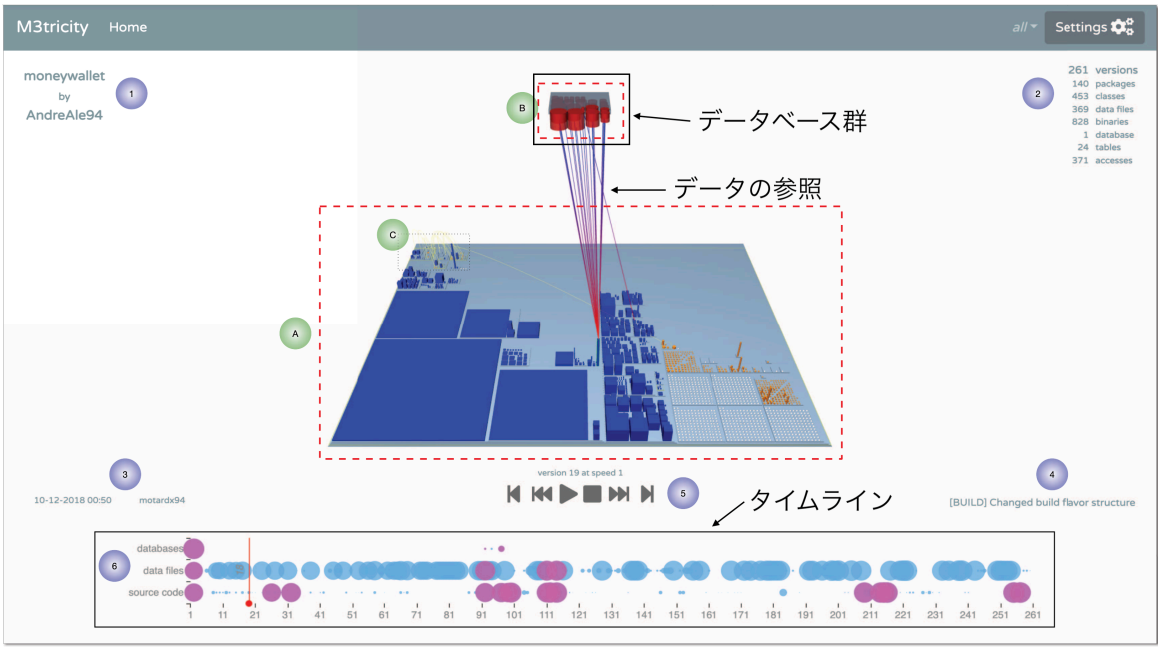

図 2　M3triCity（[10] の Figure 3 に加筆）

クラスの順に地区として表し，メソッドを建物として表す．メソッドである建物の高さは，ある固定時間におけるメソッドの実行時間の占める割合によって動的に変化するため，ユーザはパフォーマンスの時間変化を容易に認識することができる．

本研究では，ユーザの操作に対応したクライアントサイドでのリクエストの送信，およびレスポンスを受け取った際に可視化を行う．

4.3　都市メタファーの利点

Web アプリケーションは多数のコンポーネントから構成される分散システムである．分散システムの構造は複雑であるため，クライアントサイドとサーバーサイドのつながり，コンポーネント同士のつながりを理解することは困難である．

都市メタファーを利用することにより，同じ場所には存在していないクライアントサイドとサーバーサイドを都市の地区として表し，それぞれに存在するコンポーネントを建物として表すことにより，1 画面上において Web アプリケーションの構造を見ることができる．また，アプリケーション内のデータの移動を建物と建物の間を移動するトラックとして表すことで，普段の生活と密接したものに置き換えることにより，動作のつながりの理解が容易になると考える．

5.　提案システム

5.1　図書貸し出しアプリケーション

提案システムの可視化の対象として作成する図書館貸し出しアプリケーションについて説明する．作成する Web アプリケーションと既存の Web アプリケーションフレームワークとの関係性，および作成する図書館貸し出しアプリケーションが対象として妥当であることについて論ずる．

5.1.1　図書貸し出しアプリケーションの機能

図書貸し出しアプリケーションは，

- データベースへの本の追加，
- データベースにある本の削除，
- データベースにある本の全件表示，
- 本の作者名やタイトル名での検索，
- 本の貸し出し，
- 本の返却

の 6 つの機能を実現する．

5.1.2 図書貸し出しアプリケーションの構成

図書貸し出しアプリケーションの設計を図 3 に示す．図書貸し出しアプリケーションのフロントエンドを JavaScript，バックエンドを Python で作成する．フロントエンドとバックエンド間の通信には Python の Web アプリケーションフレームワークである Flask を使用する．フロントエンドではブラウザ上のユーザ入力を受け取り，XMLHttpRequest をバックエンド側に送信し，サーバーから返ってきた応答をブラウザ上に表示する．バックエンドではユーザが入力したデータをフロントエンド側から受け取り，データベースへの処理を行う．処理が終わり次第，フロントエンド側へ応答を送信する．

アプリケーション全体の設計には MVC モデルを採用する．

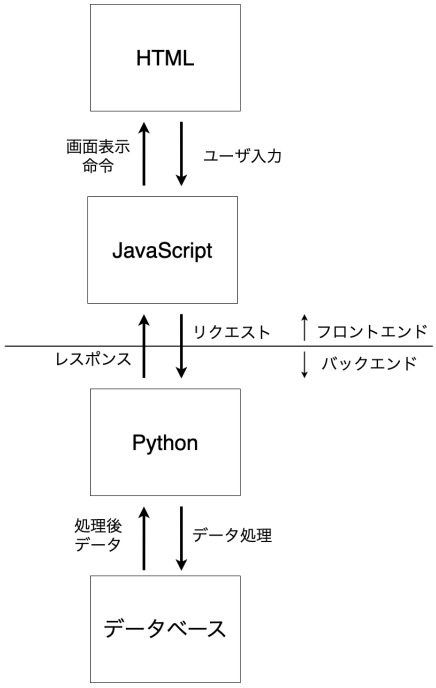

図 3 図書貸し出しアプリケーションの設計

5.1.3 適用する Web アプリケーションの妥当性

本研究の実験対象として作成する図書貸し出しアプリケーションが，普及している Web アプリケーションと同等の構造をなしていることを説明して，研究対象としてふさわしいことを以下の 3 点を用いて論ずる．

(1)Web アプリケーションの 3 層構造を成している

基本的な Web アプリケーションは，ブラウザ上に表示を行うプレゼンテーション層，ユーザの入力に基づく処理を行うアプリケーション層，データの抽出やデータの保管を行うデータ層の 3 層構造を成している．

作成する図書貸し出しアプリケーションは，ブラウザへ表示を行う HTML，ユーザの入力をバックエンドへ送信，バックエンド側からの応答を表示するよう命令を行う JavaScript からなるフロントエンド，フロントエンドから送られてきたデータに基づいてデータベースへの処理を行い，処理内容をフロントエンド側へ送るバックエンド，バックエンド側からの処理命令に基づき，図書の検索や追加を行いデータをバックエンド側に送信するデータベースによる 3 層構造をとっている．

(2) 扱うデータ形式が JSON である

フロントエンド，バックエンド間でやり取りされる，ユーザ入力の内容，それぞれで実行される関数の情報などは JSON 形式を用いる．Web アプリケーションにおけるデータ交換形式には JSON が広く使われている．

(3) 通信方法に XMLHttpRequest を用いている

フロントエンドとバックエンド間でのデータやり取りを行うための通信方法として Web API である XMLHttpRequest を用いる．HTTP プロトコルを用いたクライアントサイドとサーバーサイドの 1 対 1 対応の通信を実現している．

以上の 3 点より，図書貸し出しアプリケーションは基本的な Web アプリケーションと同等の構造，通信方法をとっているため研究の対象として用いるにふさわしいことが言える．

5.2 可視化システムについて

システムは図書貸し出しアプリケーションに，内部構造や動作の処理を可視化する機能をつける形で実装する．

システムは Web ブラウザ上で動作する．シス

テムは JavaScript で構成されており，Web アプリケーションにおけるデータの内容や，フロントエンド，バックエンドのコンポーネント内の関数処理に関する情報を取得し，動作の流れを順を追って可視化していく．本システムにおける可視化は図書貸し出しアプリケーションのウインドウとは別のウインドウで行う．

5.2.1 システムの構造

システム全体の構造を図 4 に示す．図書貸し出しアプリケーションのブラウザ表示を行う HTML と，リクエストの送信とサーバサイドからのレスポンスを受信する JavaScript との間に，可視化システムに可視化用のデータを送るための仲介役を用意する．可視化システムの仲介役と可視化システムの作成には JavaScript を作成し，可視化には JavaScript の 3D オブジェクト描画用のライブラリである Three.js を用いる．サーバーサイド側の可視化データの収集を行うために，図書貸し出しアプリケーションの Python にデータ収集部を追加する．

5.2.2 静的な可視化

ブラウザ上で，ユーザが図書貸し出しアプリケーションを立ち上げた時の静的な可視化の例を図 5 に示す．図書貸し出しアプリケーションの構造をクライアントサイド，サーバーサイドの地区に分ける．データベースはクライアントサイドとサーバーサイドの地区とは別の地区に描画する．クライアントサイドとサーバーサイドの地区の上にファイルを表す地区を設定し，ファイルの地区の上にクラスである建物を描画する．クラスである建物の高さはクラス全体のコード行数，底面積はクラス内のメソッド数で表す．クライアントサイドとサーバーサイドの通信路はクライアントサイドの地区とサーバーサイドの地区を結ぶ道として表す．

5.2.3 動的な可視化

5.2.3.1 Web アプリケーションの動作の見せ方

ユーザが入力したデータがクライアントサイドからサーバサイドまで送信され，サーバーサイドでの処理を終え，画面上に戻ってくるまでのシステムの動作は，クラスへのデータ格納，クラス内のメソッド呼び出し，それぞれのサイド，データベースを表す地区，建物へのデータ移動，データの変換に分けられる．

クラスへのデータ格納はクラスである建物を黄色く明滅させることで表現する．クラス内のメソッドの呼び出しでは，クラスである建物の上部に，メソッド名が呼ばれたことをポップアップ表示する．建物や地区へのデータ移動はトラックの移動によって表す．トラックの荷台の部分にはデータを表す直方体のオブジェクトを乗せる．データの変換は，メソッドのポップアップ表示を行った後，データを表すオブジェクトの大きさを変化させることで表す．

5.2.3.2 動的な可視化を行う際に必要なデータ

ここでは動的な可視化に必要なデータ収集の方法と可視化システムへのデータの伝達方法について説明する．

動的な可視化を行う際に必要なデータは，入力されたデータと，呼び出されるクラス名，メソッド名，処理後のデータである．動的な可視化に必要なデータは，アプリケーションが本来行うバックエンドからのレスポンスに付随してやり取りを行う．データ収集に必要なコードは図書貸し出しアプリケーションのコードに手作業で埋め込む．

6. 評価方法

6.1 実験方法

図書貸し出しアプリケーションに対して本システムを使用した上で評価指標に関するアンケートに回答してもらい，本システムの評価を行う．

6.2 評価指標

本システムの評価指標として以下の 3 つを扱う．

図書貸し出しアプリケーションの一連の処理の理解度

ブラウザ上の入力をクライアントサイドからサーバーサイドへ送信，処理後のデータがサーバサイドからクライアントサイドまで返信されブラウザ上に表示されるまでの，どのクラスのどのメソッドが使用されているかが正しく理解できたかについて 5 段階の評価を行ってもらう．

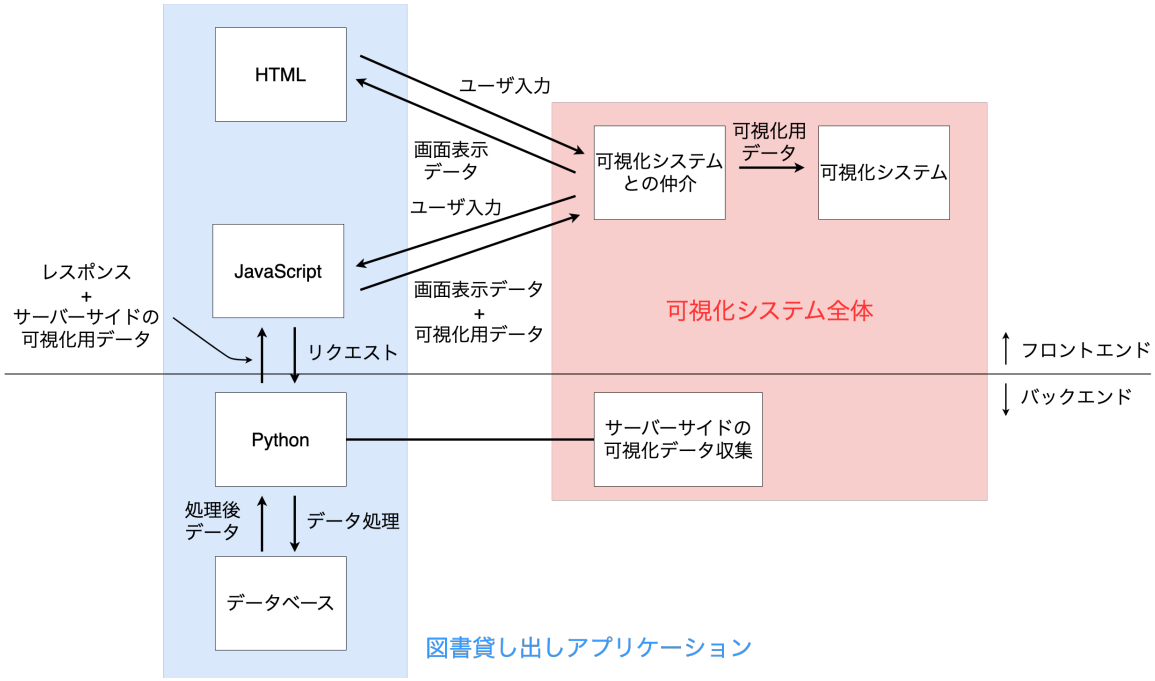

図 4　システムの概要図

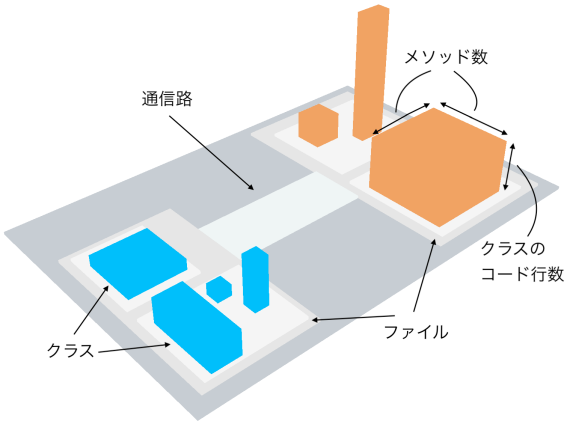

図 5　静的な可視化の画面設計
右奥側の橙色の建物がフロントエンドのクラスを，
青色の建物がバックエンドのクラスを表している．

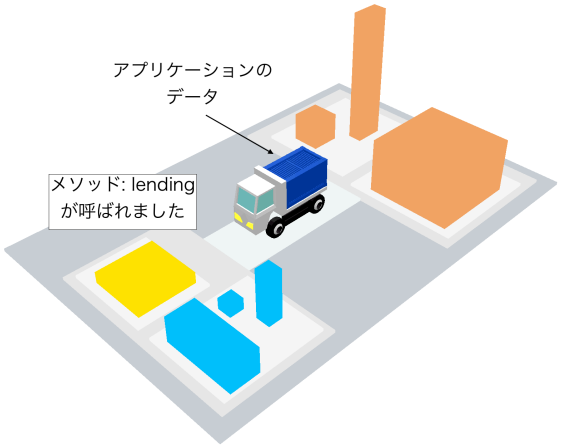

図 6　メソッド呼び出しの画面設計
左端で黄色く光っている建物であるクラス内のメソッドである lending の呼び出しが建物上部に表示される．

クラスとメソッドの特定

図書貸し出しアプリケーション内のこちらが指定したクラスやメソッドが，可視化システム内で描画されているどの建物なのか，どのクラス内で呼び出されているかに答えてもらう．

システムに関する自由記述

システムを利用してみて，使いやすさや可視化のわかりやすさなどについての自由記述を行ってもらう．

ソースコード 1　通信でやり取りされる JSON データ

```
{
    "application_data": { // 本来の図書貸し出しアプリケーションのデータ
        "title": 吾輩は猫である,
        "author": 夏目漱石,
        ...,
    },
    "visualize_data": { // 可視化に必要なデータ
        "class_name": library_operation,
        "method_name": lending,
        "args": {
            ...
        }
    }
}
```

7. 今後の課題

現在の可視化に必要なデータ収集はアプリケーションのコードにデータ収集のコードを手作業で埋め込むことで実現している．今後は可視化に必要なデータ収集のコードを自動で埋め込めるようにし，アプリケーションと可視化システムを分離することを考えている．

参考文献

[1] Google Chrome デベロッパーツール，https://www.google.com/intl/ja/chrome/dev/.

[2] J. Cloutier, S. Kpodjedo and G. El Boussaidi, “WAVI: A reverse engineering tool for web applications”, 2016 IEEE 24th International Conference on Program Comprehension (ICPC), pp. 1-3.

[3] 伊藤秀朗，団野博文，三部良太，“実行時情報に基づく Web アプリケーション可視化手法”，情報処理学会研究報告ソフトウェア工学（SE），Vol.2012-SE-177 No.11，2012.

[4] Charles, https://www.charlesproxy.com/.

[5] Flask, https://flask.palletsprojects.com/en/3.0.x/.

[6] Laravel, https://laravel.com/.

[7] Richard Wettel, Michele Lanza, “Visualizing Software Systems as Cities”, 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, pp. 92-99.

[8] Richard Wettel, “Visual exploration of large-scale evolving software”, 2009 31st International Conference on Software Engineering - Companion Volume, pp.391-394.

[9] R. Wettel, M. Lanza and R. Robbes, “Software systems as cities: a controlled experiment”, 2011 33rd International Conference on Software Engineering (ICSE), pp. 551-560.

[10] Susanna Ardigò, Csaba Nagy, Roberto Minelli, Michele Lanza, “M3triCity: Visualizing Evolving Software & Data Cities”, 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), pp.130-133.

[11] 大神勝也，中才恵太朗，畑秀明，松本健一，“Heijo: 動的なコード実行可視化による Java/Android アプリケーションのプロファイラ”，日本ソフトウェア科学会コンピュータソフトウェア，Vol.36，No.2(2019)，pp.93-105.