

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



BÀI TẬP LỚN -  
LẬP TRÌNH NÂNG CAO (CO2039)

---

# WEB TRA CỨU DỮ LIỆU SAO KÊ CỦA MTTQ VN

---

<b>Giảng viên:</b>	Thầy Lê Đình Thuận
<b>Nhóm:</b>	L01 - 501
<b>Sinh viên thực hiện:</b>	Nguyễn Thanh Hải - 2110152 Lê Trần Anh Dũng - 2210576 Nguyễn Đức Duy - 2210510 Phan Thành Long - 2211891 Võ Trường Chinh - 2320002 Bùi Trung Quân - 2312817

Tp. Hồ Chí Minh, Tháng 09/2024



## MỤC LỤC

<b>1</b>	<b>GIỚI THIỆU ĐỀ TÀI</b>	<b>4</b>
<b>2</b>	<b>PHÂN TÍCH YÊU CẦU</b>	<b>5</b>
2.1	Đề tài: Web/App sao kê . . . . .	5
2.2	Yêu cầu chức năng . . . . .	5
2.3	Kiểm thử . . . . .	5
<b>3</b>	<b>PHÂN CÔNG CÔNG VIỆC</b>	<b>6</b>
3.1	Phân tích yêu cầu và định hướng phát triển . . . . .	6
3.2	Phát triển frontend . . . . .	6
3.3	Phát triển Backend . . . . .	6
3.4	Phát triển API . . . . .	6
<b>4</b>	<b>THIẾT KẾ HỆ THỐNG</b>	<b>7</b>
4.1	Kiến trúc hệ thống tổng quan . . . . .	7
4.1.1	Mô tả . . . . .	7
4.1.2	Diagram . . . . .	7
4.2	Cấu trúc file dữ liệu . . . . .	8
4.3	Thiết kế API . . . . .	8
4.3.1	Quy ước URL và HTTP method . . . . .	8
4.3.2	Định dạng dữ liệu . . . . .	8
<b>5</b>	<b>TÌM HIỂU CÔNG NGHỆ SỬ DỤNG</b>	<b>9</b>
5.1	Phát triển Backend . . . . .	9
5.1.1	Springboot . . . . .	9
5.2	Phát triển Frontend . . . . .	11
5.3	Xây dựng API . . . . .	11
5.3.1	Giới thiệu về RESTful API . . . . .	11
5.3.2	Các thành phần cốt lõi của RESTful API . . . . .	11
<b>6</b>	<b>PHÁT TRIỂN BACKEND</b>	<b>13</b>
6.1	Công nghệ sử dụng . . . . .	13
6.1.1	Giới thiệu . . . . .	13
6.1.2	Phân tích các công nghệ sử dụng . . . . .	13
6.2	Các module chức năng . . . . .	15
6.2.1	Mô tả chi tiết về các file và chức năng . . . . .	15
6.3	Cấu trúc dữ liệu và giải thuật được sử dụng . . . . .	17
6.3.1	Cấu trúc dữ liệu . . . . .	17
6.3.2	Giải thuật . . . . .	17

<b>7</b>	<b>PHÁT TRIỂN FRONTEND</b>	<b>19</b>
7.1	Mục đích của dự án . . . . .	19
7.2	Tính năng của hệ thống . . . . .	19
7.3	Công nghệ sử dụng . . . . .	19
7.3.1	React, HTML, CSS . . . . .	19
7.3.2	FetchAPI . . . . .	20
7.3.3	@workday/canvas-kit-react/pagination . . . . .	20
7.4	Phân tích chi tiết mã nguồn . . . . .	20
7.4.1	Xử lý tìm kiếm dữ liệu . . . . .	20
7.4.2	Làm nổi bật từ khóa tìm kiếm . . . . .	21
7.4.3	Phân trang dữ liệu . . . . .	21
<b>8</b>	<b>TÍCH HỢP VÀ KIỂM THỬ</b>	<b>23</b>
8.1	Kiểm thử hệ thống . . . . .	23
8.2	Kiểm thử giao diện người dùng . . . . .	25
<b>9</b>	<b>TRIỂN KHAI VẬN HÀNH</b>	<b>29</b>
9.1	Hướng dẫn triển khai . . . . .	29
9.1.1	Frontend (FE) . . . . .	29
9.1.2	Backend (BE) . . . . .	29
<b>10</b>	<b>TÍNH NĂNG CÓ THỂ MỞ RỘNG TRONG TƯƠNG LAI</b>	<b>30</b>
10.1	Các Tính năng có thể mở rộng trong tương lai . . . . .	30
10.1.1	Hỗ trợ nhiều tiêu chí tìm kiếm nâng cao . . . . .	30
10.1.2	Tích hợp các tính năng xuất dữ liệu . . . . .	30
10.1.3	Hỗ trợ ngôn ngữ khi tìm kiếm . . . . .	30



## DANH SÁCH HÌNH VẼ

4.1	Diagram . . . . .	7
5.1	Logo của Spring Boot. . . . .	9
8.1	Kiểm thử hệ thống . . . . .	23
8.2	Giao diện tìm kiếm theo thời gian (Date_time). . . . .	26
8.3	Giao diện tìm kiếm theo số thứ tự (Trans_no). . . . .	26
8.4	Giao diện tìm kiếm theo số tiền ghi có (Credit). . . . .	27
8.5	Giao diện tìm kiếm theo số tiền ghi nợ (Debit). . . . .	27
8.6	Giao diện tìm kiếm theo nội dung giao dịch (Detail). . . . .	28

## CHƯƠNG 1

### GIỚI THIỆU ĐỀ TÀI

**Giới thiệu:** Sao kê giao dịch là nhu cầu thiết yếu để theo dõi và quản lý thông tin tài chính. Đề tài này hướng đến việc xây dựng một ứng dụng hỗ trợ tra cứu sao kê giao dịch của Mặt trận Tổ quốc Việt Nam. Ứng dụng được thiết kế dưới dạng web/app hoặc giao diện dòng lệnh (CLI), cho phép người dùng dễ dàng tìm kiếm thông tin giao dịch dựa trên số tiền, tên người gửi và nội dung giao dịch.

**Mục tiêu:** Ứng dụng sẽ hỗ trợ tra cứu thông tin giao dịch theo các tiêu chí như số tiền giao dịch, tên người gửi và nội dung giao dịch. Đồng thời, ứng dụng được thiết kế để hiển thị thông tin từ tệp dữ liệu nguồn một cách nhanh chóng và hiệu quả.

**Dữ liệu đầu vào:** Dữ liệu sử dụng trong dự án được cung cấp [tại đây](#). Tệp dữ liệu này chứa thông tin chi tiết về các giao dịch, bao gồm số tiền, tên người gửi và nội dung giao dịch.

**Kiến trúc hệ thống:** Ứng dụng sẽ bao gồm hai thành phần chính. Thứ nhất, giao diện người dùng sẽ hỗ trợ truy vấn thông tin từ tệp dữ liệu đầu vào theo các tiêu chí tìm kiếm, đồng thời hiển thị kết quả dưới dạng bảng hoặc danh sách. Thứ hai, xử lý tìm kiếm sẽ tích hợp các giải thuật tìm kiếm tối ưu để xử lý và hiển thị dữ liệu dựa trên các tiêu chí số tiền, tên người gửi và nội dung giao dịch.

**Kết luận:** Đề tài không chỉ hướng đến việc xây dựng một công cụ hỗ trợ tìm kiếm sao kê mà còn rèn luyện kỹ năng thiết kế hệ thống, tối ưu hóa xử lý dữ liệu và tạo ra một ứng dụng hiệu quả, thân thiện với người dùng.

## CHƯƠNG 2

## PHÂN TÍCH YÊU CẦU

### 2.1 Đề tài: Web/App sao kê

**Nhóm:** 05 - 07 SV

**Mô tả:** Xây dựng một web/app Tra cứu dữ liệu “Sao Kê” của MTTQ VN.

**Datasource:** [https://s.thuanle.me/chuyen\\_khoan.csv](https://s.thuanle.me/chuyen_khoan.csv)

### 2.2 Yêu cầu chức năng

- Có giao diện (web/app/CLI) để có thể tra cứu thông tin sao kê theo:
  - Số tiền
  - Tên người gửi
  - Nội dung
- **Giới hạn:** Sinh viên có thể sử dụng các thư viện bên thứ 3 để hỗ trợ xây dựng các cấu trúc dữ liệu cho việc tìm kiếm, nhưng không được sử dụng 1 hệ thống/ứng dụng dịch vụ độc lập xử lý toàn bộ việc tìm kiếm (ví dụ như không được dùng DB để chứa dữ liệu và chỉ query tìm kiếm một cách out-of-the-box).

### 2.3 Kiểm thử

- **Nội bài:** Source, slide
- **Thuyết trình:** Về cấu trúc dữ liệu/giải thuật đã xây dựng cho việc tìm kiếm.
- **Quy định điểm:**
  - Điểm thang max cho đề tài bình thường là 9đ.
  - Load test/stress test: 1đ (Dành cho các nhóm có xây dựng Web App/có api Web).
  - Sẽ benchmark dịch vụ của các nhóm để nhận 1 điểm thưởng còn lại.
- **Công cụ dự định dùng để benchmark:**
  - <https://k6.io/>
  - <https://github.com/tsenart/vegeta>
- **Endpoint dùng để perform test:** <http://goku:9090/query?q=search> (giả sử host là goku:9090)

## CHƯƠNG 3

### PHÂN CÔNG CÔNG VIỆC

Trong dự án, nhiệm vụ được phân công cụ thể để đảm bảo công việc được thực hiện hiệu quả và đúng tiến độ. Các công việc chính như sau:

#### 3.1 Phân tích yêu cầu và định hướng phát triển

- Lê Trần Anh Dũng - 2210576
- Nguyễn Thanh Hải - 2110152
- Nguyễn Đức Duy - 2210510

#### 3.2 Phát triển frontend

- Nguyễn Đức Duy - 2210510
- Võ Trường Chinh - 2320002
- Phan Thành Long - 2211891

#### 3.3 Phát triển Backend

- Lê Trần Anh Dũng - 2210576
- Nguyễn Thanh Hải - 2110152

#### 3.4 Phát triển API

- Bùi Trung Quân - 2312817

## CHƯƠNG 4

## THIẾT KẾ HỆ THỐNG

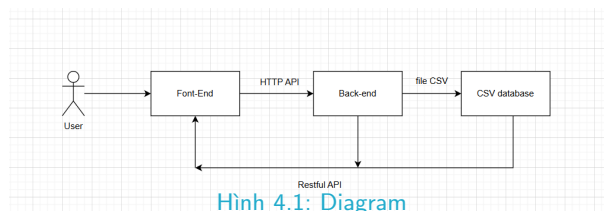
### 4.1 Kiến trúc hệ thống tổng quan

#### 4.1.1 Mô tả

Hệ thống này được thiết kế để cho phép người dùng truy vấn và lấy dữ liệu từ một nguồn thông qua giao diện người dùng (Frontend - FE), xử lý dữ liệu trên máy chủ (Backend - BE), và trả kết quả cho người dùng thông qua API Restful. Quy trình này sử dụng các bước sau:

- **Frontend (FE):** Giao diện người dùng sẽ gửi yêu cầu API đến Backend thông qua phương thức HTTP để truy xuất dữ liệu.
- **Backend (BE):** Máy chủ (backend) sẽ nhận yêu cầu từ frontend và tiếp nhận các thông tin cần thiết (từ một file CSV). Dữ liệu từ file CSV sẽ được lấy ra, xử lý hoặc tìm kiếm, rồi trả về cho frontend qua API.
- **CSV:** Là nguồn dữ liệu chính trong hệ thống. File CSV chứa dữ liệu cần thiết, mà backend sẽ đọc và trích xuất các thông tin từ đó để cung cấp cho frontend.
- **Restful API:** Backend sẽ triển khai một API theo chuẩn RESTful để giao tiếp với frontend. API này có thể hỗ trợ nhiều phương thức (GET) để thực hiện truy vấn dữ liệu.

#### 4.1.2 Diagram



Hình 4.1: Diagram

Giải thích sơ đồ:

- Frontend (FE): Là giao diện người dùng, có thể là một ứng dụng web hoặc mobile. Frontend sẽ gửi yêu cầu HTTP đến Backend để truy vấn dữ liệu.
- Backend (BE): Máy chủ xử lý các yêu cầu từ frontend, quản lý các hoạt động như truy xuất, xử lý dữ liệu từ nguồn CSV. Backend sẽ trả dữ liệu qua API Restful.
- CSV Database: Là nơi lưu trữ dữ liệu. File CSV chứa thông tin mà Backend cần xử lý hoặc tìm kiếm để gửi cho frontend.



- Restful API: Giao tiếp giữa frontend và backend sử dụng API chuẩn RESTful, thường là thông qua các định dạng dữ liệu như json.

## 4.2 Cấu trúc file dữ liệu

Mỗi dòng trong file CSV đại diện cho một giao dịch và chứa các thông tin như sau:

- **date\_time:** Chuỗi thể hiện thời điểm xảy ra giao dịch, với định dạng ngày/tháng/năm\_giây. Trường này cung cấp thời gian chính xác của giao dịch.
- **trans\_no:** Số nguyên đại diện cho mã số duy nhất của giao dịch. Mỗi giao dịch sẽ có một mã số khác nhau để dễ dàng theo dõi và phân biệt.
- **credit** và **debit:** Hai trường này ghi nhận số tiền được ghi có vào (credit) hoặc rút ra (debit) từ tài khoản trong giao dịch. Cả hai đều có kiểu dữ liệu số thực.
- **detail:** Chuỗi chứa thông tin chi tiết về giao dịch, bao gồm mã giao dịch, nội dung giao dịch và có thể thêm thông tin về người gửi hoặc người nhận tiền.

Ví dụ về một dòng trong file CSV có thể trông như sau: Nguồn dữ liệu là `chuyen_khoan.csv`, với

```
1["date_time","trans_no","credit","debit","detail"]
2"01/09/2024_5215.97152",1,"3000",0,"01/09/2024_5215.97152"
3"01/09/2024_5219.24714",2,"10000",0,"010800.010924.223139.chuc mung ngay 29 nước cộng hòa xã hội chủ nghĩa Việt Nam"
4"02/09/2024_5212.22905",3,"10000",0,"055464.820924.064157.ung ho nha nước Việt Nam và Glup do Nguoi dan (by TPBank ChatPay)"
```

khoảng 20,000 dòng, chứa  $n$  trường dữ liệu giống như mô tả ở trên. Mỗi dòng trong file này đại diện cho một giao dịch, cung cấp thông tin về thời điểm, mã số giao dịch, số tiền ghi có hoặc rút ra, và chi tiết về giao dịch.

## 4.3 Thiết kế API

### 4.3.1 Quy ước URL và HTTP method

Quy ước URL: `http://localhost:8080/search?attribute=date_time&data=10`

Phương thức HTTP: GET

### 4.3.2 Định dạng dữ liệu

Dữ liệu trả về là JSON mặc định với 10 kết quả mỗi trang.

```
{
  "data": [
    "\"03/09/2024 5212.961101,12,100000,0, 160707.030924.112000. NGUYEN HOAI PHUONG CK BCT TW",
    "\"07/09/2024 9915.102931,128,9,155000 THU PHI DỊCH VỤ SMS CHỦ DONG THANG 68/2024. SDT: 096432163",
    "\"07/09/2024 156156 tat ho sol 5217.97166,156,\"10000\",6,1927805.070924.150156 hoan 64/89/2024 5",
    "\"05/09/2024 5244,10447,96,\"20000\",6,\"HBVCB 6955998468.NGUYEN HUYEN TRANG chuyen tien. CT tu",
    "\"07/09/2024 5212. 16610,157,120001\",0.1641603.079924.150924. IBFT BUI THỊ HƯỜNG chuyen tien\"",
    "88/09/2024 5390.10351,189,14360009,02009704050908013609202401VY045159.1835 1.013609. Vietcombank",
    "\"06/09/2024 5017.10731,203,\"6000\",0,\"627713,680924.101824, HOANG THỊ CHI Chuyen tien",
    "\"08/09/2024 5220,42710,240, 3000, 0,033039.088924.153455, HOÀNG THỊ NGAN chuyen tien",
    "08/09/2024 5246.218811,252,1500060\",,\"MBVCB 6976925698. TRẦN NGUYEN THANH TUYEN UNG HO KHÁC PHU",
  ],
  "totalPages": 19368,
  "totalRecords": 193676
}
```

## CHƯƠNG 5

# TÌM HIỂU CÔNG NGHỆ SỬ DỤNG

### 5.1 Phát triển Backend

#### 5.1.1 Springboot



Hình 5.1: Logo của Spring Boot.

**Giới thiệu:** Spring Boot là một framework mạnh mẽ dựa trên Spring Framework, được thiết kế để đơn giản hóa quá trình phát triển ứng dụng backend. Ra đời vào năm 2014, Spring Boot giúp giảm thiểu cấu hình thủ công thông qua việc tự động hóa các thiết lập và cung cấp cấu hình mặc định.

**Ưu điểm:**

- Tích hợp sẵn các máy chủ nhúng như Tomcat, Jetty.
- Hỗ trợ xây dựng RESTful API và ứng dụng web dễ dàng với các starter.
- Tự động hóa cấu hình, tăng tốc độ phát triển ứng dụng.
- Cung cấp công cụ giám sát và quản lý ứng dụng thông qua Spring Boot Actuator.

##### 5.1.1.1 Spring Boot Starter Parent

**Chức năng:** Là một parent POM, cung cấp cấu hình sẵn từ Spring Boot, giúp quản lý dependencies hiệu quả hơn.

**Phiên bản 3.0.0:**

- Hỗ trợ Java 17.
- Cập nhật dependencies mới nhất, đảm bảo sự ổn định và bảo mật.

#### 5.1.1.2 Spring Boot Starter Web

**Chức năng:**

- Hỗ trợ xây dựng ứng dụng web và RESTful API.
- Tích hợp sẵn Spring MVC và máy chủ nhúng Tomcat.

**Lợi ích:**

- Giảm thiểu cấu hình khi tạo API.
- Tối ưu hóa hiệu năng cho các ứng dụng web backend.

#### 5.1.1.3 Spring Boot Starter Test

**Chức năng:** Cung cấp các công cụ kiểm thử mạnh mẽ, bao gồm:

- **JUnit:** Viết unit test cho các thành phần Java.
- **Mockito:** Hỗ trợ mocking đối tượng trong kiểm thử.

**Ứng dụng:** Thực hiện unit test và integration test để đảm bảo chất lượng mã nguồn.

#### 5.1.1.4 Công nghệ xử lý tệp CSV

**Apache Commons CSV:**

- Hỗ trợ đọc và ghi tệp CSV hiệu quả.
- Thích hợp cho xử lý dữ liệu dạng bảng hoặc xuất/nhập dữ liệu.

**OpenCSV:**

- Mở rộng khả năng ánh xạ dữ liệu CSV vào các đối tượng Java (Beans).
- Hỗ trợ xử lý các tệp phức tạp với định dạng đa dạng.

#### 5.1.1.5 Swagger và OpenAPI

**springfox-boot-starter:**

- Tích hợp Swagger 2, tự động tạo tài liệu REST API.
- Cung cấp giao diện Swagger UI, cho phép kiểm thử API trực tiếp trên trình duyệt.

**springdoc-openapi-ui:**

- Hỗ trợ OpenAPI 3.0, cung cấp tài liệu API chi tiết và rõ ràng hơn.
- Cho phép xuất tài liệu dưới dạng JSON/YAML, dễ dàng tích hợp với các công cụ khác.

#### 5.1.1.6 Spring Boot Maven Plugin

**Chức năng:**

- Hỗ trợ build và chạy ứng dụng trực tiếp từ Maven.
- Đóng gói ứng dụng thành tệp .jar hoặc .war.

**Lợi ích:**

- Đơn giản hóa quy trình triển khai.
- Tăng tốc độ phát triển nhờ quy trình tự động hóa.

## 5.2 Phát triển Frontend

Công nghệ **React**, **HTML**, **CSS** được sử dụng để xây dựng giao diện người dùng (UI) động, mang lại trải nghiệm mượt mà và tương tác linh hoạt cho người dùng. React sử dụng kiến trúc dựa trên các thành phần (components), trong đó mỗi phần của giao diện được chia thành các thành phần độc lập, dễ dàng tái sử dụng và quản lý. **State management** (quản lý trạng thái) được sử dụng để quản lý dữ liệu động trong giao diện, như các kết quả đã lọc hoặc trang hiện tại, giúp UI phản ứng linh hoạt với các thay đổi trong dữ liệu.

Fetch API được sử dụng để kết nối frontend và backend, gửi các yêu cầu HTTP nhằm lấy dữ liệu từ server, chẳng hạn như yêu cầu tìm kiếm của người dùng. Một ví dụ code có thể là:

Thư viện **@workday/canvas-kit-react/pagination** giúp tạo tính năng phân trang cho bảng dữ liệu, giúp người dùng dễ dàng duyệt qua các trang kết quả mà không bị quá tải thông tin. Thư viện này tự động quản lý số lượng dữ liệu hiển thị trên mỗi trang và hiển thị chi tiết tổng phân trang với các kiểu cột dữ liệu như Ngày tháng, Tín dụng và Nợ, giúp người dùng dễ dàng theo dõi và quản lý thông tin.

**React** được phát triển bởi Facebook (hiện tại thuộc Meta) và lần đầu tiên được phát hành vào năm 2013. Công nghệ này nhanh chóng trở thành một trong những thư viện JavaScript phổ biến nhất nhờ vào tính mô-đun cao, khả năng tái sử dụng components và khả năng tương tác mạnh mẽ với người dùng. Kể từ đó, React đã trải qua nhiều phiên bản cập nhật, với các tính năng mới như hooks và context API, giúp cải thiện hiệu suất và sự linh hoạt trong việc phát triển ứng dụng web.

## 5.3 Xây dựng API

### 5.3.1 Giới thiệu về RESTful API

Trong thời đại công nghệ số hiện nay, việc kết nối và trao đổi dữ liệu giữa các ứng dụng ngày càng trở nên quan trọng. RESTful API đã nổi lên như một tiêu chuẩn phổ biến và hiệu quả cho việc giao tiếp giữa các hệ thống.

#### RESTful API là gì?

Hãy tưởng tượng một nhà hàng. Bạn (ứng dụng) là khách hàng, nhà bếp là hệ thống xử lý dữ liệu, và người phục vụ chính là RESTful API. Bạn đặt món (yêu cầu dữ liệu) với người phục vụ, người phục vụ sẽ chuyển yêu cầu đến nhà bếp và mang món ăn (dữ liệu) trở lại cho bạn.

RESTful API hoạt động theo cách tương tự, đóng vai trò là cầu nối giúp các ứng dụng gửi yêu cầu dữ liệu đến hệ thống khác. Đặc biệt, RESTful API sử dụng các phương thức HTTP quen thuộc (GET, POST, PUT, DELETE) và các định dạng dữ liệu phổ biến như JSON hoặc XML để truyền tải thông tin một cách rõ ràng và nhất quán.

**Ví dụ:** Khi bạn sử dụng Facebook trên điện thoại, ứng dụng này sẽ sử dụng RESTful API để “gọi món” dữ liệu từ server của Facebook, chẳng hạn như lấy danh sách bạn bè, hiển thị bài viết mới nhất, hoặc đăng một trạng thái mới.

#### Lý do nên sử dụng RESTful API cho đề tài này:

- **Đơn giản và dễ hiểu:** RESTful API tuân thủ các nguyên tắc thiết kế đơn giản, dễ nắm bắt, giúp việc phát triển API cho ứng dụng sao kê trở nên thuận tiện hơn.
- **Linh hoạt và tương thích:** RESTful API có thể được sử dụng với nhiều ngôn ngữ lập trình và nền tảng khác nhau (web, app, CLI), phù hợp với yêu cầu đề tài.
- **Khả năng mở rộng:** RESTful API có thể dễ dàng mở rộng để đáp ứng nhu cầu tra cứu dữ liệu sao kê ngày càng tăng.
- **Hiệu quả:** RESTful API thường sử dụng ít băng thông hơn so với các kiểu API khác, giúp tối ưu hóa hiệu suất cho ứng dụng.

### 5.3.2 Các thành phần cốt lõi của RESTful API

- **Tài nguyên (Resource):** Trong đề tài này, tài nguyên chính là dữ liệu "Sao kê". Mỗi bản ghi sao kê có thể được coi là một tài nguyên và được xác định bởi một URL duy nhất. Ví dụ, `/transactions/123` đại diện cho bản ghi sao kê có ID là 123.

- **Phương thức HTTP:** Ứng dụng sẽ sử dụng các phương thức HTTP để thực hiện các thao tác trên dữ liệu sao kê:
  - **GET:** Lấy thông tin sao kê. Ví dụ, `GET /transactions` để lấy danh sách tất cả các bản ghi sao kê, hoặc `GET /transactions?q=người gửi A` để tìm kiếm các bản ghi có người gửi là "người gửi A".
  - **POST:** Tạo mới bản ghi sao kê (nếu được yêu cầu). Ví dụ, `POST /transactions` để thêm một bản ghi sao kê mới.
- **Định dạng dữ liệu:** Sử dụng JSON để biểu diễn dữ liệu sao kê. Ví dụ, khi lấy thông tin một bản ghi với `GET /transactions/123`, API có thể trả về dữ liệu dạng JSON như sau:

```
{  
  "id": 123,  
  "người gửi": "Nguyễn Văn A",  
  "người nhận": "MTTQ VN",  
  "số tiền": 100000,  
  "nội dung": "Ủng hộ quỹ vì người nghèo"  
}
```

## CHƯƠNG 6

## PHÁT TRIỂN BACKEND

### 6.1 Công nghệ sử dụng

#### 6.1.1 Giới thiệu

Đoạn mã trong bài báo cáo được viết bằng ngôn ngữ lập trình Java, sử dụng các công nghệ Java Core kết hợp với Spring Framework. Chức năng chính của đoạn mã là tìm kiếm dữ liệu từ tệp CSV dựa trên một số thuộc tính đầu vào, đồng thời tích hợp tính năng phân trang để quản lý kết quả trả về một cách hiệu quả.

Bài báo cáo này phân tích các công nghệ được sử dụng trong đoạn mã, làm rõ vai trò và cách thức triển khai của chúng.

#### 6.1.2 Phân tích các công nghệ sử dụng

##### 1. Java Core

##### 1.1. Xử lý tệp (File Handling)

- Công nghệ:
  - Lớp `BufferedReader` và `FileReader` từ thư viện Java IO được sử dụng để đọc dữ liệu từ tệp CSV.
- Chi tiết kỹ thuật:
  - `FileReader` mở tệp CSV và đọc ký tự từng dòng.
  - `BufferedReader` sử dụng bộ nhớ đệm để đọc dữ liệu một cách hiệu quả hơn.
  - Cú pháp `try-with-resources` đảm bảo tệp sẽ được tự động đóng sau khi xử lý, giảm thiểu nguy cơ rò rỉ tài nguyên:

```
1 try (BufferedReader br = new BufferedReader(new FileReader(FILE_PATH))) {
```

- Dòng đầu tiên của tệp CSV được đọc riêng để lấy tiêu đề (header), hỗ trợ xác định các cột.

##### 1.2. Xử lý chuỗi (String Handling)

- Các phương thức như `split()`, `toLowerCase()`, và `contains()` được sử dụng để xử lý và kiểm tra dữ liệu:
  - `split(",")`: Chia một chuỗi thành các phần tử nhỏ dựa trên dấu phẩy.
  - `toLowerCase()`: Chuyển đổi chuỗi sang chữ thường để thực hiện tìm kiếm không phân biệt chữ hoa/chữ thường.
  - `contains()`: Kiểm tra xem một chuỗi con có tồn tại trong chuỗi lớn hơn hay không.

### 1.3. Xử lý số (Numeric Handling)

- Đối với các cột chứa giá trị số (như credit và debit), phương thức `Integer.parseInt()` được sử dụng để chuyển đổi chuỗi thành số nguyên. Nếu giá trị không hợp lệ (ví dụ, chứa ký tự không phải số), một ngoại lệ `NumberFormatException` sẽ được bắt.

## 2. Spring Framework

### 2.1. Annotation @Service

- Lớp `CSVSearchService` được đánh dấu với annotation `@Service`, cho phép Spring Framework quản lý đối tượng này như một "service".
- Annotation này giúp tổ chức mã nguồn rõ ràng, đồng thời hỗ trợ Spring tự động khởi tạo và quản lý vòng đời của đối tượng.

### 2.2. Tính năng tích hợp

- Việc sử dụng Spring giúp đoạn mã dễ dàng tích hợp vào các ứng dụng lớn hơn, đặc biệt trong kiến trúc phân tầng (layered architecture), nơi lớp Service thường xử lý logic nghiệp vụ.

## 3. Tính Năng Phân Trang (Pagination)

### 3.1. Ý nghĩa

- Phân trang là một tính năng quan trọng khi xử lý danh sách dữ liệu lớn. Đoạn mã sử dụng hai tham số page và size để giới hạn số lượng bản ghi được trả về.

### 3.2. Triển khai

- Công thức tính toán:
  - `fromIndex = page * size`: Vị trí bắt đầu của dữ liệu trên trang hiện tại.
  - `toIndex = Math.min(fromIndex + size, results.size())`: Vị trí kết thúc của dữ liệu trên trang hiện tại, không vượt quá tổng số bản ghi.
- Kết quả được cắt (sub-list) từ danh sách đầy đủ dựa trên phạm vi từ `fromIndex` đến `toIndex`.

### 3.3. Trả về kết quả

- Kết quả phân trang được đóng gói trong đối tượng `PageResponse`, bao gồm:
  - Danh sách kết quả (results).
  - Tổng số trang (totalPages).
  - Tổng số bản ghi (totalRecords).

## 4. Xử Lý Ngoại Lệ (Exception Handling)

### 4.1. IOException

- Phát sinh khi tệp không tồn tại hoặc không thể đọc được.
- Được xử lý bằng khối lệnh `try-catch`, đảm bảo chương trình không bị dừng đột ngột:

```
} catch (IOException e) {  
    e.printStackTrace();  
}  
4
```

### 4.2. NumberFormatException

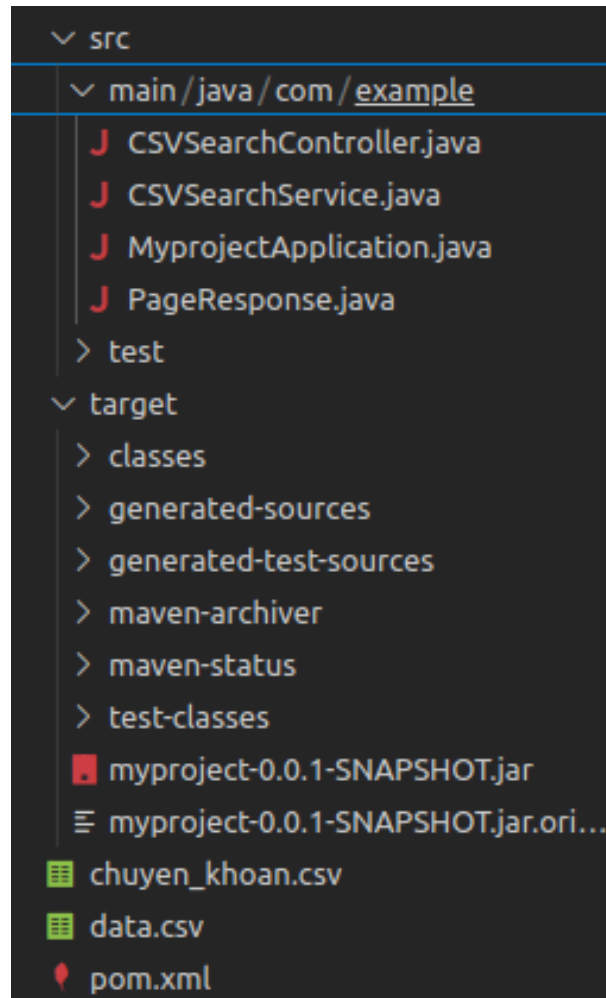
- Phát sinh khi chuyển đổi chuỗi thành số không hợp lệ, ví dụ: chuỗi "abc" trong cột yêu cầu dữ liệu số.
- Được xử lý nhẹ nhàng bằng cách bỏ qua dòng dữ liệu lỗi:

```
} catch (NumberFormatException e) {  
2    // Bỏ qua lỗi  
}  
}
```

## 6.2 Các module chức năng

### 6.2.1 Mô tả chi tiết về các file và chức năng

#### 6.2.1.1 Cấu trúc thư mục và các file Java



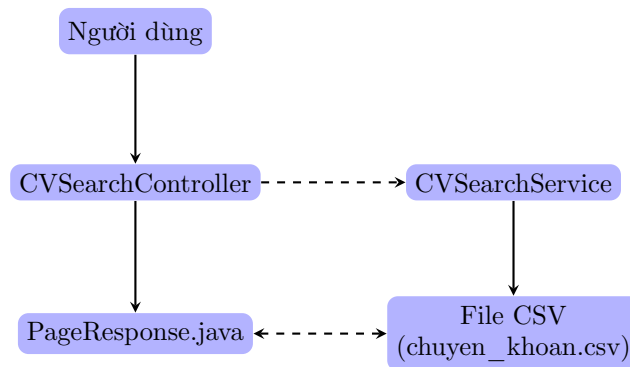
- **Thư mục /src/main/java/com/example:** Đây là thư mục gốc chứa các lớp Java cho dự án.
- **Các file Java:**
  - **CSVSearchController.java:**
    - \* **Chức năng:** Đây là lớp controller, đóng vai trò như một điểm tiếp xúc giữa ứng dụng và bên ngoài (thường là một frontend). Nó sẽ nhận các yêu cầu tìm kiếm từ bên ngoài, xử lý các yêu cầu đó và trả về kết quả.
    - \* **Ví dụ:** Khi một người dùng gửi một yêu cầu tìm kiếm trên một trang web, yêu cầu này sẽ được gửi đến controller này. Controller sẽ trích xuất thông tin từ yêu cầu (ví dụ: từ khóa, số trang, số lượng kết quả trên một trang) và gọi đến service để thực hiện tìm kiếm.
  - **CSVSearchService.java:**
    - \* **Chức năng:** Đây là lớp service, chứa logic thực tế để tìm kiếm trong file CSV. Nó sẽ nhận các thông tin tìm kiếm từ controller, thực hiện tìm kiếm trong file CSV và trả về kết quả.
    - \* **Chi tiết:**
      - Đọc file CSV: Mở file CSV, đọc từng dòng và chia nhỏ từng dòng thành các cột.



- Tìm kiếm: Tìm kiếm theo cột và giá trị đã cho. Hỗ trợ tìm kiếm số và chuỗi.
  - Phân trang: Chia kết quả tìm kiếm thành các trang, mỗi trang có một số lượng kết quả nhất định.
- **MyprojectApplication.java:**
- \* **Chức năng:** Đây là lớp chính của ứng dụng, thường chứa phương thức `main` để khởi động ứng dụng. Nó sẽ cấu hình các bean, component và các phần khác của ứng dụng.
- **PageResponse.java:**
- \* **Chức năng:** Đây là một lớp đơn giản để đóng gói kết quả trả về của phương thức tìm kiếm. Nó thường chứa:
    - Danh sách các dòng tìm thấy.
    - Tổng số trang kết quả.
    - Tổng số bản ghi tìm thấy.

#### 6.2.1.2 Sơ đồ hoạt động

1. **Người dùng:** Gửi một yêu cầu tìm kiếm đến ứng dụng (ví dụ: qua một trang web).
2. **Controller:** Nhận yêu cầu, trích xuất thông tin và gọi đến service.
3. **Service:**
  - Đọc file CSV.
  - Thực hiện tìm kiếm.
  - Phân trang kết quả.
  - Trả về kết quả cho controller.
4. **Controller:** Nhận kết quả từ service và trả về cho người dùng (ví dụ: dưới dạng JSON).



#### 6.2.1.3 Tổng kết

- **Mục đích:** Ứng dụng này được thiết kế để tìm kiếm thông tin trong một file CSV.
- **Các thành phần:**
  - Controller: Nhận yêu cầu và trả về kết quả.
  - Service: Thực hiện logic tìm kiếm.
  - Model: Đóng gói dữ liệu (PageResponse).
- **Ưu điểm:**
  - Cấu trúc rõ ràng, dễ hiểu.
  - Hỗ trợ phân trang.
  - Có thể tùy chỉnh để tìm kiếm nhiều loại dữ liệu khác nhau.

Lưu ý:

- **File CSV:** File "chuyen\_khoan.csv" chứa dữ liệu cần tìm kiếm. Cấu trúc của file này sẽ ảnh hưởng đến cách thức tìm kiếm.
- **Phân trang:** Phân trang giúp cải thiện hiệu suất khi làm việc với các file CSV lớn.
- **Khả năng mở rộng:** Ứng dụng này có thể được mở rộng để hỗ trợ nhiều loại file khác nhau (ví dụ: Excel, JSON), các loại tìm kiếm phức tạp hơn (ví dụ: tìm kiếm theo nhiều tiêu chí, tìm kiếm theo khoảng).

## 6.3 Cấu trúc dữ liệu và giải thuật được sử dụng

### 6.3.1 Cấu trúc dữ liệu

Trong đoạn mã, các cấu trúc dữ liệu chính được sử dụng bao gồm:

- **List<String>:**
  - *Mục đích:* Lưu trữ các dòng dữ liệu thỏa mãn điều kiện tìm kiếm.
  - *Sử dụng:* Biến `results` dùng để lưu kết quả.
- **String:**
  - *Mục đích:* Lưu trữ dữ liệu từng dòng, tên tiêu đề, hoặc các giá trị cột để so sánh.
  - *Sử dụng:*
    - \* Dùng để đọc từng dòng từ tệp CSV (`line`).
    - \* Lưu trữ tiêu đề cột (`headerLine`) gán nó vào `column index`.
- **Mảng (String[]):**
  - *Mục đích:* Lưu trữ các giá trị trong một dòng CSV sau khi tách bằng dấu phẩy.
  - *Ví dụ sử dụng:*
    - \* `headers` lưu các tiêu đề cột từ dòng đầu tiên.
    - \* `values` lưu các giá trị của dòng dữ liệu hiện tại.

### 6.3.2 Giải thuật

Vì file dữ liệu chỉ khoảng 20 nghìn dòng nên chúng em dùng giải thuật linear Search để tiếp cận nhằm tìm kiếm kết quả với độ phức tạp  $O(n.m)$  cho tìm kiếm và  $O(p)$  cho phân trang

#### 6.3.2.1 Bước 1: Đọc tệp CSV

- Mở tệp `chuyen_khoan.csv` bằng `BufferedReader`.
- Đọc dòng đầu tiên để lấy tiêu đề cột (`headerLine`).
- Nếu tệp rỗng hoặc không có tiêu đề, trả về kết quả rỗng.

#### 6.3.2.2 Bước 2: Xác định cột cần tìm kiếm

- Tách tiêu đề thành mảng `headers` bằng phương thức `split(",")`.
- Duyệt qua mảng `headers` để tìm vị trí (`columnIndex`) của cột tương ứng với thuộc tính đầu vào (`attribute`).
- Nếu không tìm thấy cột, trả về kết quả rỗng.

### 6.3.2.3 Bước 3: Lọc dữ liệu

- Đọc từng dòng dữ liệu còn lại trong tệp.
- Với mỗi dòng:
  - Tách các giá trị thành mảng `values`.
  - Kiểm tra điều kiện:
    - \* Nếu cột là số (`columnIndex == 1 || columnIndex == 2`), chuyển giá trị và dữ liệu đầu vào sang số nguyên để so sánh.
    - \* Nếu cột là chuỗi, kiểm tra xem giá trị có chứa dữ liệu đầu vào hay không (không phân biệt chữ hoa/thường).
  - Nếu điều kiện thỏa mãn, thêm dòng vào danh sách `results`.

### 6.3.2.4 Bước 4: Phân trang

- Tính tổng số bản ghi thỏa mãn (`totalRecords`) và tổng số trang (`totalPages`).
- Tính chỉ số bắt đầu (`fromIndex`) và kết thúc (`toIndex`) của các dòng thuộc trang hiện tại.
- Trả về danh sách con (`results.subList(fromIndex, toIndex)`) cùng với thông tin phân trang.

## CHƯƠNG 7

## PHÁT TRIỂN FRONTEND

### 7.1 Mục đích của dự án

Mục tiêu chính của dự án là xây dựng một hệ thống trực tuyến cho phép người dùng nhập liệu và tra cứu dữ liệu giao dịch một cách dễ dàng. Hệ thống này được thiết kế để giúp:

- Cá nhân và tập thể quản lý và theo dõi thông tin về việc ủng hộ đồng bào miền trung để khắc phục thiệt hại của cơn bão yagi
- Giảm tải công việc xử lý giấy tờ và tiết kiệm thời gian trong việc tìm kiếm thông tin giao dịch.

### 7.2 Tính năng của hệ thống

Hệ thống bao gồm các tính năng chính:

- Tìm kiếm và lọc dữ liệu theo nhiều tiêu chí: ngày tháng(Date time), số tiền(Credit), số thứ tự(Train no), Số ghi nợ(Debit) và nội dung tìm kiếm(Detail).
- Hiển thị kết quả theo thời gian thực với giao diện trực quan
- Phân trang dữ liệu và làm nổi bật các từ khóa tìm kiếm
- Tương tác với backend để xử lý dữ liệu lớn và trả kết quả nhanh chóng.

### 7.3 Công nghệ sử dụng

#### 7.3.1 React, HTLM, CSS

- **Mục đích:** Xây dựng giao diện người dùng (UI) động.
- **Đặc điểm nổi bật:**
  - Component-based architecture: Mỗi phần của giao diện được chia thành các thành phần độc lập, dễ quản lý.
  - State management: Sử dụng state để quản lý dữ liệu động, chẳng hạn như filteredResults và currentPage.

### 7.3.2 FetchAPI

- **Mục Đích:** Nhóm sử dụng Fetch API để thực hiện kết nối giữa frontend và backend. Thực hiện các yêu cầu tìm kiếm của người dùng.

- **VD code:**

```
const response = await fetch(`/search?attribute=${searchType}&data=${searchValue}&page=${currentPage}`);
```

### 7.3.3 @workday/canvas-kit-react/pagination

Mục đích: Tạo bản phân trang (pagination) cho bảng dữ liệu.

Đặc Điểm:

- Tự động quản lý số lượng hiển thị dữ liệu trên mỗi trang
- Hiển thị chi tiết tổng phân trang với các kiểu cột dữ liệu như Date, Credit, Debit,...

## 7.4 Phân tích chi tiết mã nguồn

### 7.4.1 Xử lý tìm kiếm dữ liệu

Dưới đây là một đoạn mã JavaScript thực hiện chức năng tìm kiếm:

```
const handleSearch = async () => {  
  2    try {  
        const response = await fetch(  
          `/search?attribute=${searchType}&data=${searchValue}&page=${currentPage - 1}`  
        );  
        const data = await response.json();  
        7    setResults(data);  
    } catch (error) {  
        console.error("Error fetching data:", error);  
    }  
};
```

- **Mục đích:** Hàm `handleSearch` được sử dụng để thực hiện yêu cầu tìm kiếm từ người dùng, gửi dữ liệu qua Fetch API đến backend, nhận kết quả, và cập nhật vào giao diện.

- **Chi tiết hoạt động:**

1. `searchType` và `searchValue` là các tham số đầu vào của người dùng, dùng để xác định tiêu chí tìm kiếm và giá trị của tiêu chí đó.
2. `await`: Được sử dụng để đợi kết quả từ `fetch()` mà không làm gián đoạn chương trình, giúp hệ thống không bị "đóng băng" trong khi chờ dữ liệu từ server.
3. Sử dụng `fetch()` để gửi yêu cầu HTTP GET với các tham số được truyền động.
4. Chuyển đổi phản hồi từ dạng JSON sang đối tượng JavaScript thông qua `response.json()`.
5. Gọi hàm `setResults` để cập nhật kết quả vào trạng thái giao diện.
6. Xử lý lỗi bằng khối `try-catch`, đảm bảo rằng các lỗi trong quá trình fetch được log ra console.

Ý nghĩa: Cách tiếp cận này đảm bảo hiệu quả xử lý tìm kiếm, giảm thiểu thời gian chờ đợi của người dùng và cho phép tương tác trực tiếp với dữ liệu từ server. Bằng việc sử dụng `async/await`, mã nguồn trở nên dễ đọc và duy trì hơn.

## 7.4.2 Làm nổi bật từ khóa tìm kiếm

Dưới đây là một đoạn mã JavaScript thực hiện chức năng tìm kiếm:

```
const highlightKeyword = ( text , keyword ) => {  
  const regex = new RegExp ( keyword , 'gi' );  
  return text . replace ( regex , match => < span    class =" highlight " >  
    $ { match } </ span > ) ;  
};
```

- `RegExp(keyword, 'gi')`: Tạo một đối tượng biểu thức chính quy (RegExp) để tìm kiếm tất cả các từ khóa trong chuỗi văn bản text. Tham số g giúp tìm tất cả các kết quả, và i cho phép tìm kiếm không phân biệt chữ hoa/thường.
- `text.replace(regex, ...)`: Phương thức này thay thế tất cả các kết quả tìm được (từ khóa) bằng một thẻ `<span>` có class `highlight`, giúp làm nổi bật chúng trên giao diện người dùng.
- Hiển thị nội dung sau khi làm nổi bật

```
    return text.replace(regex, (match)=> `<span class="highlight">${match}</span>`);
```

## 7.4.3 Phân trang dữ liệu

```
<tbody>  
  {filteredResults.map((item, index) => (  
    <tr key={index}>  
      {renderCell(item, "date_time")}  
      {renderCell(item, "trans_no")}  
      {renderCell(item, "credit")}  
      {renderCell(item, "debit")}  
      {renderCell(item, "detail")}  
    </tr>  
  ) )}  
</tbody>  
<Pagination  
  currentPage={currentPage} // Truyn currentPage vo Pagination  
  onPageChange={handlePageChange} // Gi hm khi chuyn trang  
  aria-label="Pagination"  
  lastPage={lastPage}  
  className="custom-pagination"  
</>  
{/* Nt chuyn v trang u */}  
<button  
  aria-label="Go to First Page"  
  className="page-nav-button"  
  onClick={() => handlePageChange(1)} // Chuyn v trang u  
  disabled={currentPage === 1} // V hui ha nu ang trang u tin  
>  
  &laquo; First  
</button>  
{/* Nt gim trang */}  
<button  
  aria-label="Previous Page"  
  className="page-nav-button"  
  onClick={() => {  
    if (currentPage > 1) handlePageChange(currentPage - 1);  
  }}  
  disabled={currentPage === 1} // V hui ha nu ang trang u tin  
>  
  &lt;
```

```

</button>
{/* Danh sách các trang */}
40 {range.map((pageNumber) => (
  <Pagination.PageListItem key={pageNumber} className="page-item">
    <Pagination.PageButton
      aria-label={`Page ${pageNumber}`}
      pageNumber={pageNumber}
45      className={`page-button ${pageNumber === currentPage ? "active" : ""}`}
      onClick={() => handlePageChange(pageNumber)}
    >
      {pageNumber}
    </Pagination.PageButton>
50 </Pagination.PageListItem>
  )
)}
{/* Nút trang cuối */}
<button
  aria-label="Next Page"
55  className="page-nav-button"
  onClick={() => {
    if (currentPage < totalPages) handlePageChange(currentPage + 1);
  }}
  disabled={currentPage === totalPages} // Vô hiệu hóa nút trang cuối
60 >
  &gt;
</button>
{/* Nút chuyển về trang cuối */}
<button
65  aria-label="Go to Last Page"
  className="page-nav-button"
  onClick={() => handlePageChange(totalPages)} // Chuyển về trang cuối
  disabled={currentPage === totalPages} // Vô hiệu hóa nút trang cuối
  >
70  Last &raquo;
</button>

```

## 1. Hiển thị dữ liệu:

- Sử dụng `filteredResults.map()` để hiển thị các dòng dữ liệu trong bảng (`<tr>`)
- `renderCell(item, field)` là hàm hiển thị từng cột dữ liệu như `datetime`, `transno`, `credit`, ...

## 2. Cấu trúc phân trang:

- Nút chuyển về trang đầu
  - Hàm `handlePageChange(1)` được gọi để quay về trang đầu tiên.
  - Nút bị vô hiệu hóa (`disabled`) nếu đã ở trang đầu.
- Nút giảm/trang trước: Giảm trang hiện tại xuống 1 nếu chưa ở trang đầu.
- Danh sách số trang:
  - `range` là một mảng chứa các số trang.
  - Số trang hiện tại được tô sáng với class `active`.
- Nút tăng/trang sau: Tăng trang hiện tại lên 1 nếu chưa ở trang cuối.
- Nút chuyển về trang cuối: Hàm `handlePageChange(totalPages)` chuyển trực tiếp về trang cuối.

## CHƯƠNG 8

## TÍCH HỢP VÀ KIỂM THỬ

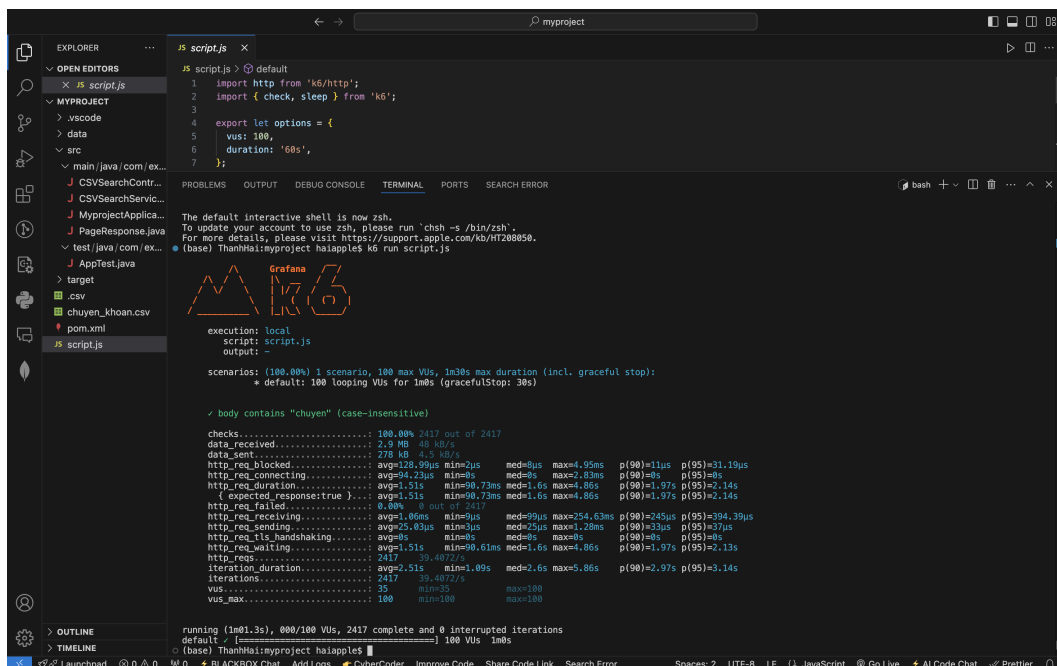
### 8.1 Kiểm thử hệ thống

#### 1. Kiểm thử trên máy:

- Tên máy: Macbook Air M1
- Chip: Apple M1
- MacOS: Sonoma 14.6.1
- RAM: 16GB

#### 2. Kết quả:

Trường hợp: 100 request với sleep 1s



```
1 import http from 'k6/http';
2 import { check, sleep } from 'k6';
3
4 export let options = {
5   vus: 100,
6   duration: '60s',
7 };
8
9 // body contains "chuyen" (case-insensitive)
10 checks: 100.00% 2417 out of 2417
11 data_received: 2.9 MB 48 kB/s
12 data_sent: 278 KB 4.5 kB/s
13 http_req_blocked: avg=128.09µs min=0µs med=0µs max=2.83ms p(90)=11µs p(95)=31.19µs
14 http_req_connecting: avg=94.23µs min=0µs med=0µs max=2.83ms p(90)=8µs p(95)=8µs
15 http_req_duration: avg=1.51s min=90.73ms med=1.0s max=4.86s p(90)=1.97s p(95)=2.14s
16 http_req_failed: 0.00% 0 out of 2417
17 http_req_receiving: avg=1.06ms min=0µs med=99µs max=254.63ms p(90)=245µs p(95)=394.39µs
18 http_req_sending: avg=25.83µs min=0µs med=25µs max=1.28ms p(90)=13µs p(95)=37µs
19 http_req_tls_handshaking: avg=0µs min=0µs med=0µs max=0µs p(90)=0µs p(95)=0µs
20 http_req_waiting: avg=1.51s min=90.61ms med=1.0s max=4.86s p(90)=1.97s p(95)=2.13s
21 http_res_body_size: avg=1.51s min=90.61ms med=1.0s max=4.86s p(90)=1.97s p(95)=2.13s
22 iteration_duration: avg=2.51s min=1.09s med=2.0s max=5.86s p(90)=2.97s p(95)=3.14s
23 iterations: 2417 39.4872/s
24 vus: 100 min=35 max=100
25 vus_max: 100 min=100 max=100
26
27 running (1m01.3s), 000/100 VUs, 2417 complete and 0 interrupted iterations
28 default: [ 100 VUs ] 100 VUs 1mb
```

Hình 8.1: Kiểm thử hệ thống

#### 3. Phân tích:



### 3.1. Thông tin chung:

#### 3.1.1. Công cụ kiểm tra:

- Công cụ test: k6.
- Tích hợp giám sát qua Grafana.

#### 3.1.2. Phương thức thực thi:

- Execution: Local.
- Script: script.js (tập lệnh kiểm tra được chạy).
- Output: Không ghi ra file, kết quả chỉ hiển thị trực tiếp.

#### 3.1.3. Kịch bản thử nghiệm:

- Có 1 kịch bản: default.
- Số lượng người dùng ảo (VUs): Tối đa 100.
- Thời gian chạy: 1 phút chính thức và 30 giây dừng mượt (graceful stop).

### 3.2. Kết quả chính:

#### 3.2.1. Tổng số vòng lặp hoàn thành (iterations):

- 2417 ( 39.41 vòng/s).

#### 3.2.2. Kiểm tra nội dung:

- Điều kiện: Phản hồi HTTP chứa chuỗi "chuyen"(không phân biệt chữ hoa/chữ thường).
- Tỷ lệ kiểm tra đạt yêu cầu (checks): 100% (2417/2417).

### 3.3. Hiệu suất hệ thống:

#### 3.3.1. Dữ liệu trao đổi:

- Dữ liệu nhận (data\_received): 2.9 MB ( 48 KB/s).
- Dữ liệu gửi (data\_sent): 278 KB ( 4.5 KB/s).

#### 3.3.2. Thời gian thực hiện HTTP Requests:

##### a. Thời gian tổng hợp (http\_req\_duration):

- Trung bình: 1.51s.
- Tối thiểu: 90.73ms.
- Tối đa: 4.86s.
- Phân vị (percentile):
  - 90th (p90): 1.97s.
  - 95th (p95): 2.14s.

##### b. Các giai đoạn của request:

- Blocked: (thời gian chờ trước khi gửi request):
  - Trung bình: 128.99µs.
  - Tối đa: 4.95ms.
- Connecting: (thời gian kết nối TCP):
  - Trung bình: 94.23µs (ổn định, không có thời gian cao).
- Waiting: (thời gian server xử lý):
  - Trung bình: 1.51s.
- Receiving: (thời gian nhận dữ liệu):
  - Trung bình: 1.06ms.
- Sending: (thời gian gửi dữ liệu):
  - Trung bình: 25.03µs.

##### c. Tỷ lệ thất bại (http\_req\_failed):

- 0% (Không có request nào bị lỗi).

#### 3.3.3. Thời gian vòng lặp:

- Thời gian trung bình mỗi vòng lặp: 2.51s.
- Tối thiểu: 1.09s.
- Tối đa: 5.86s.

- Phân vị:
  - 90th (p90): 2.97s.
  - 95th (p95): 3.14s.

#### 3.3.4. Virtual Users (VUs):

- VUs thực tế: Dao động từ 35 đến 100 trong suốt quá trình.
- VUs tối đa: 100.

### 3.4. Đánh giá:

#### 3.4.1. Hiệu năng:

- Hệ thống xử lý ổn định dưới tải tối đa 100 VUs.
- Không có request thất bại (`http_req_failed` = 0%).
- Thời gian phản hồi trung bình 1.51s là chấp nhận được nhưng có thể cần tối ưu cho trải nghiệm tốt hơn (giảm thời gian waiting).

#### 3.4.2. Độ ổn định mạng:

- Thời gian kết nối (connecting) và blocked rất thấp, cho thấy mạng ổn định.
- Tốc độ gửi/nhận dữ liệu ổn định và phù hợp với tải.

#### 3.4.3. Khả năng đáp ứng nội dung:

- Tất cả response đều chứa chuỗi "chuyen", đạt yêu cầu.

## 8.2 Kiểm thử giao diện người dùng

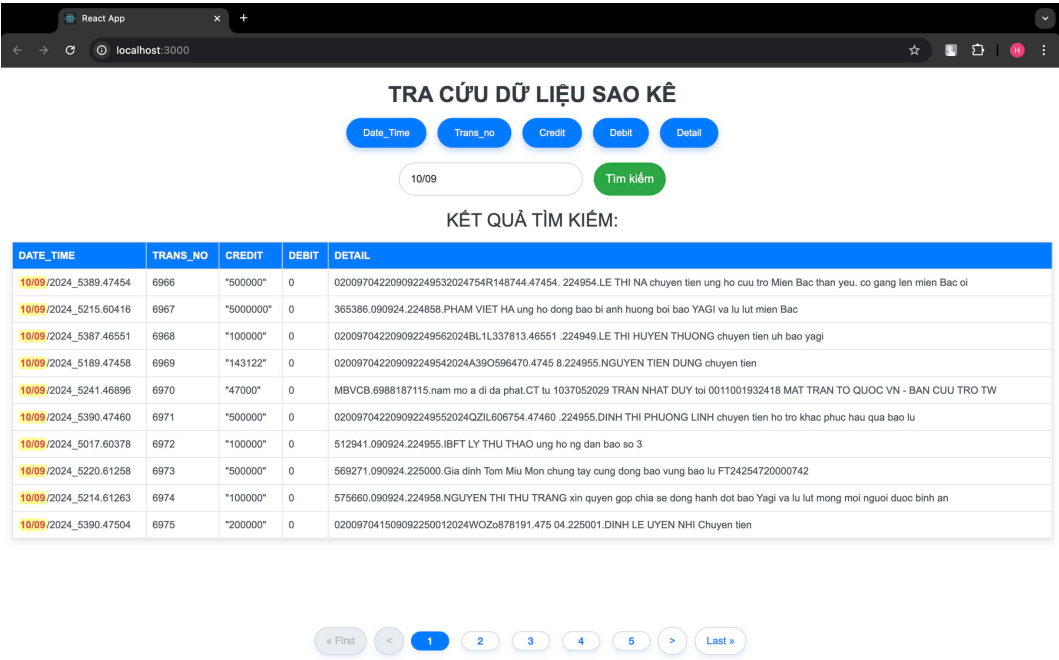
Cách chạy chương trình font end:

1. Thực hiện download Nodejs
2. Dùng lệnh: `npm start` thực hiện chạy chương trình

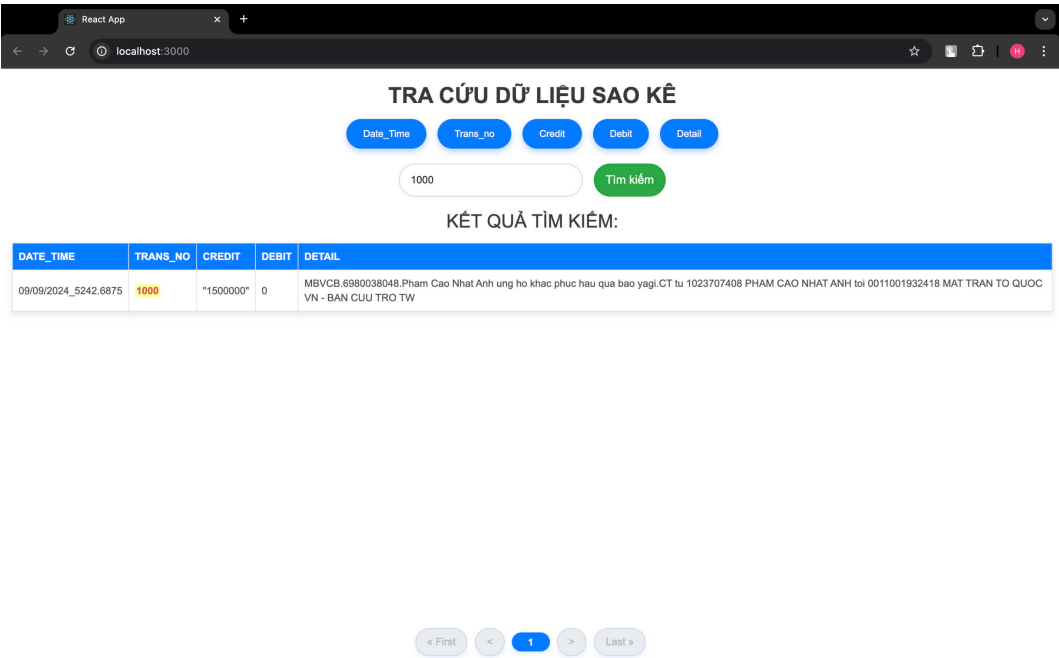
Nhóm thực hiện tìm kiếm với 5 kiểu tìm kiếm lần lượt là :

- **Tìm kiếm theo thời gian (Date):** Sử dụng định dạng **Ngày/Tháng/Năm**. Ví dụ: 10/09/2024.
- **Tìm kiếm theo số thứ tự giao dịch (Trans\_no):** Sử dụng định dạng **number**.
- **Tìm kiếm theo số tiền ghi có (Credit):** Sử dụng định dạng **number**. Ví dụ: 1000.
- **Tìm kiếm theo số tiền ghi nợ (Debit):** Sử dụng định dạng **number**. Ví dụ: 0.
- **Tìm kiếm theo nội dung giao dịch (Detail):** Sử dụng định dạng **text**. Ví dụ: chuyen.

Dưới đây là hình ảnh minh họa việc tìm kiếm.



Hình 8.2: Giao diện tìm kiếm theo thời gian (Date\_time).



Hình 8.3: Giao diện tìm kiếm theo số thứ tự (Trans\_no).

React App

localhost:3000

TRA CỨU DỮ LIỆU SAO KÊ

Date\_Time

Trans\_no

Credit

Debit

Detail

1000

Tìm kiếm

KẾT QUẢ TÌM KIẾM:

DATE_TIME	TRANS_NO	CREDIT	DEBIT	DETAIL
08/09/2024_5078.65740	392	*1000*	0	MBVCB.6977766225.Khokhan.CT tu 0441000722147 toi 0011001932418 Uy Ban Trung uong Mat tran To quoc Viet Nam
08/09/2024_5078.9696	658	*1000*	0	MBVCB.6979123351.LE MINH DUC chuyen tien.CT tu 1015919694 LE MINH DUC toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
09/09/2024_5244.33450	5758	*1000*	0	MBVCB.6987713025.ungho.CT tu 9824331995 LUONG TRUNG GIANG toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5078.55121	47782	*1000*	0	MBVCB.6991865969.The Ngoc que 74 ung ho thiet hai do bao so 3.CT tu 1025049194 NGUYEN THE NGOC toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5245.65959	55424	*1000*	0	MBVCB.69922179951.NGUYEN SON HAI DUC chuyen tien.CT tu 9918858685 NGUYEN SON HAI DUC toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5078.67459	55887	*1000*	0	MBVCB.6992207714.NGUYEN SON HAI DUC chuyen tien.CT tu 9918858685 NGUYEN SON HAI DUC toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5243.66786	56083	*1000*	0	MBVCB.6992211998.NGUYEN VAN THAI chuyen tien.CT tu 1016806432 NGUYEN VAN THAI toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5242.67272	56457	*1000*	0	MBVCB.6992220545.NGUYEN SON HAI DUC chuyen tien.CT tu 9918858685 NGUYEN SON HAI DUC toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5242.70454	58642	*1000*	0	MBVCB.6992328539.cuu tro lu luu.CT tu 064100032465 VO NGOC TAN toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW
10/09/2024_5242.82641	67049	*1000*	0	MBVCB.6992691376.VU TRAN KHANH HUY chuyen tien.CT tu 1041003546 DANG CAM KHANH LY toi 0011001932418 MAT TRAN TO QUOC VN - BAN CUU TRO TW

« First

<

1

2

3

4

5

>

Last »

Hình 8.4: Giao diện tìm kiếm theo số tiền ghi có (Credit).

React App

localhost:3000

TRA CỨU DỮ LIỆU SAO KÊ

Date\_Time

Trans\_no

Credit

Debit

Detail

0

Tìm kiếm

KẾT QUẢ TÌM KIẾM:

DATE_TIME	TRANS_NO	CREDIT	DEBIT	DETAIL
01/09/2024_5215.97152	1	*3000*	0	01/09/2024_5215.97152
01/09/2024_5219.24714	2	*10000*	0	018806.010924.213139.chuc mung ngay 29 nuoc cong hoa xa hoi chu nghĩa Viet Nam
02/09/2024_5212.22965	3	*100000*	0	055464.020924.064157.Ung Ho Nha Nuoс Viet Nam Va Giup do Ngươi dân (bү TPBank ChatPay)
02/09/2024_5245.21394	4	*10000*	0	MBVCB.6924605040.gia dinh Dung Thuy Giang chuc to quoc khoe manh
02/09/2024_5078.73943	5	*50000*	0	MBVCB.6925071164.mung ngay quoc khanh.CT tu 1028808193 toi 0011001932418 Uy Ban Trung uong Mat tran To quoc Viet Nam
02/09/2024_5216.8149	6	*2000*	0	524322.020924.175952.2091945
02/09/2024_5390.64042	7	*500000*	0	0200970415090220272520240Ppe432198.64042_202714.NGUYEN THI LAN HANH Quyen gop
03/09/2024_5218.45892	8	*10000*	0	881384.030924.070324.Ung Ho Nha Nuoс Viet Nam Va Giup do Ngươi dân
03/09/2024_5216.65140	9	*200000*	0	120167.030924.100642.NGUYEN HOAI NAM ung ho
03/09/2024_5219.76834	10	*300000*	0	069690.030924.111352.Ung ho dong bao Viet Nam FT24248787369079

« First

<

1

2

3

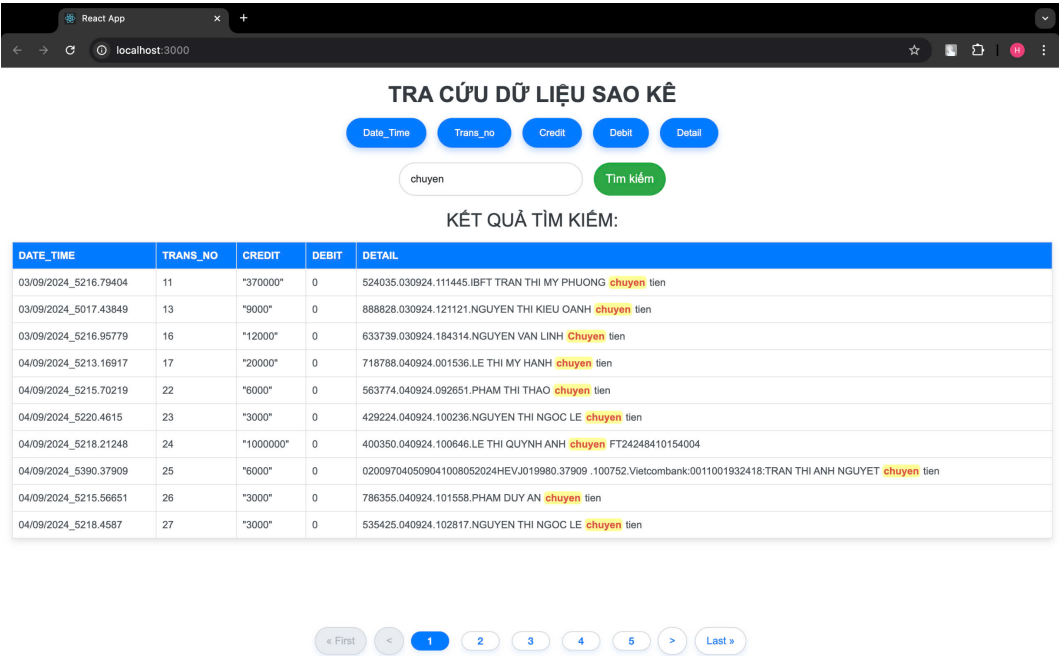
4

5

>

Last »

Hình 8.5: Giao diện tìm kiếm theo số tiền ghi nợ (Debit).



Hình 8.6: Giao diện tìm kiếm theo nội dung giao dịch (Detail).

## CHƯƠNG 9

## TRIỂN KHAI VẬN HÀNH

### 9.1 Hướng dẫn triển khai

#### 9.1.1 Frontend (FE)

Môi trường:

Node.js

Các bước triển khai:

1. **Clone mã nguồn:** Sử dụng lệnh `git clone` để clone mã nguồn frontend từ repository [https://github.com/dunghcmut/Web\\_api.git](https://github.com/dunghcmut/Web_api.git).
2. **Cài đặt dependencies:** Chạy lệnh `npm install` để cài đặt các thư viện cần thiết cho ứng dụng.
3. **Cấu hình:** Thực hiện cấu hình môi trường (nếu có yêu cầu, ví dụ như cài đặt biến môi trường hoặc tệp cấu hình).
4. **Build ứng dụng:** Chạy lệnh `npm run build` để build ứng dụng frontend.
5. **Triển khai:** Triển khai ứng dụng frontend đã build lên local host bằng cách chạy lệnh `npm start`.

#### 9.1.2 Backend (BE)

Môi trường:

Java 17 trở lên

Các bước triển khai:

1. **Chạy ứng dụng:** Sử dụng lệnh sau để chạy file JAR:

```
java -jar '/BE_sao_ke/target/myproject-0.0.1-SNAPSHOT.jar'
```

## CHƯƠNG 10

# TÍNH NĂNG CÓ THỂ MỞ RỘNG TRONG TƯƠNG LAI

### 10.1 Các Tính năng có thể mở rộng trong tương lai

#### 10.1.1 Hỗ trợ nhiều tiêu chí tìm kiếm nâng cao

- **Mô tả:** Bổ sung các tính năng tìm kiếm khác, tìm kiếm trong một khoảng thời gian và một khoảng giá tiền.
- **Lợi ích:** Giúp người dùng có thể tìm kiếm nội dung một cách chính xác và có nhiều lựa chọn để tìm kiếm hơn

#### 10.1.2 Tích hợp các tính năng xuất dữ liệu

- **Mô tả:** Xuất dữ liệu sau khi thực hiện tìm kiếm bằng các định dạng phổ biến như .csv, excel...
- **Lợi ích:** Hỗ trợ người dùng lưu trữ và chia sẻ thông tin một cách dễ dàng hơn cho các việc báo cáo.

#### 10.1.3 Hỗ trợ ngôn ngữ khi tìm kiếm

- **Mô tả:** Tích hợp nhiều loại ngôn ngữ khác nhau ( Tiếng Anh, Trung Quốc, Pháp,...).
- **Lợi ích:** Có thể đa dạng người tìm kiếm. Người tìm kiếm không bắt buộc phải dùng tiếng việt để kiểm tra mở rộng phạm vi sử dụng ra quốc tế.