**TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐẠI HỌC ĐÀ NẴNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PBL4: DỰ ÁN HỆ ĐIỀU HÀNH**  
**& MẠNG MÁY TÍNH**

**Đề tài:**

**Xây dựng chương trình Client/Server để in lịch thế kỷ**

**SINH VIÊN THỰC HIỆN:**

**Nguyễn Hữu Tùng LỚP: 21T\_DT**

**Nguyễn Ngọc Tiệp LỚP: 21T\_DT**

**GIẢNG VIÊN HƯỚNG DẪN:**

**ThS. Nguyễn Văn Nguyên**

**Đà Nẵng 11/2023**

MỤC LỤC

[1](#_Toc149942247)

[MỤC LỤC 2](#_Toc149942248)

[DANH SÁCH HÌNH VẼ 3](#_Toc149942249)

[DANH SÁCH BẢNG BIỂU 4](#_Toc149942250)

[DANH SÁCH CÁC TỪ VIẾT TẮT 5](#_Toc149942251)

[GIỚI THIỆU ĐỀ TÀI 6](#_Toc149942252)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 7](#_Toc149942253)

[1.1. Yêu cầu đề tài 7](#_Toc149942254)

[1.2. Giới thiệu mô hình Client/Server 7](#_Toc149942255)

[1.2.1. Cấu trúc mô hình Client/Server 7](#_Toc149942256)

[1.2.2. Máy chủ (Server) 7](#_Toc149942257)

[1.2.3. Máy trạm (Client) 7](#_Toc149942258)

[1.2.4. Quy trình hoạt động 7](#_Toc149942259)

[*1.3.* Giới thiệu giao thức *TCP/IP* 8](#_Toc149942260)

[1.3.1. Giới thiệu về TCP 8](#_Toc149942261)

[1.3.2. Giới thiệu về IP 8](#_Toc149942262)

[1.3.3. Giới thiệu về cổng(Port) 8](#_Toc149942263)

[1.4. Giới thiệu giao thức Http/Https 9](#_Toc149942264)

[1.4.1. Giao thức Http 9](#_Toc149942265)

[1.4.2. Giao thức Https 10](#_Toc149942266)

[1.5. Giới thiệu về các công nghệ sử dụng trong đề tài 10](#_Toc149942267)

[1.5.1. Giới thiệu về ReactJS 10](#_Toc149942268)

[1.5.2. Giới thiệu về NodeJS 12](#_Toc149942269)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 15](#_Toc149942270)

[2.1. Phân tích chức năng của chương trình 15](#_Toc149942272)

[2.2. Phân tích thiết kế của chương trình 15](#_Toc149942273)

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 16](#_Toc149942274)

[3.1. Các công cụ sử dụng 16](#_Toc149942276)

[3.2. Kết quả 16](#_Toc149942277)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 17](#_Toc149942278)

[TÀI LIỆU THAM KHẢO 18](#_Toc149942279)

[PHỤ LỤC 19](#_Toc149942280)

DANH SÁCH HÌNH VẼ

No table of figures entries found

DANH SÁCH BẢNG BIỂU

No table of table entries found.

DANH SÁCH CÁC TỪ VIẾT TẮT

No table of abbreviation entries found.

GIỚI THIỆU ĐỀ TÀI

Trong thời đại hiện nay, công nghệ thông tin đang trở thành một lực lượng mạnh mẽ, không ngừng thay đổi và cải tiến cuộc sống của chúng ta. Trong bối cảnh này, mô hình học dựa trên dự án (PBL) đã trở thành một phương pháp giảng dạy hiệu quả, đặc biệt là khi chúng ta đối mặt với thách thức giải quyết vấn đề thực tế.

Chúng em, nhóm nghiên cứu, đã chọn đề tài "Dự án viết chương trình client/server để in lịch thế kỷ" nhằm mục đích phát triển một hệ thống độc đáo, có khả năng tạo và in lịch lâu dài cho các thế kỷ khác nhau. Mục tiêu của chúng em không chỉ là tạo ra một ứng dụng chức năng, mà còn là khám phá cách tích hợp công nghệ và sáng tạo để giải quyết nhu cầu thực tế trong xã hội ngày nay.

Khả năng tiện lợi và linh hoạt của ứng dụng không chỉ giúp người dùng duy trì sự tổ chức trong cuộc sống hàng ngày mà còn mở ra những cơ hội mới trong việc tận dụng thông tin lịch sử và văn hóa. Dự án không chỉ đặt ra câu hỏi về cách chúng ta lưu giữ di sản, mà còn thách thức chúng ta suy nghĩ về cách chúng ta tương tác với thời gian hàng ngày.

Bằng cách này, chúng em hy vọng đưa ra một giải pháp tích cực cho cộng đồng, khám phá cách mà công nghệ có thể giúp cho cuộc sống hàng ngày của chúng ta trở nên dễ dàng hơn.

# CƠ SỞ LÝ THUYẾT

## Yêu cầu đề tài

Xây dựng chương trình client-server in lịch thế kỷ sử dụng javascript. Khi người dùng ở các client nhập một năm muốn in lịch thì chương trình Server thực tính toán và trả về kết quả cho client lịch của năm đó.

## Giới thiệu mô hình Client/Server

### Cấu trúc mô hình Client/Server

Client / Server là mô hình mạng máy tính gồm có 2 thành phần chính đó là máy khách (client) và máy chủ (server). Server chính là nơi giúp lưu trữ tài nguyên cũng như cài đặt các chương trình dịch vụ theo đúng như yêu cầu của client. Ngược lại, Client bao gồm máy tính cũng như các loại thiết bị điện tử nói chung sẽ tiến hành gửi yêu cầu đến server.

Mô hình mạng Client / Server sẽ cho phép mạng tập trung các ứng dụng có cùng chức năng tại một hoặc nhiều dịch vụ file chuyên dụng. Chúng sẽ trở thành trung tâm của hệ thống. Hệ điều hành của mô hình Client/Server sẽ cho phép người dùng chia sẻ đồng thời cùng một loại tài nguyên mà không giới hạn vị trí địa lý.

Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn nhất định, và chuẩn đó được gọi là giao thức. (Giao thức là gì thì các bạn có thể tham khảo tại đây) Một số giao thức được sử dụng phổ biến hiện nay như: HTTPS, TCP/IP, FTP,...

### Máy chủ (Server)

Server được hiểu là máy chủ, thường là một hệ thống máy lớn, có bộ xử lý mạnh, có khả năng hoạt động đáng tin cậy, có khả năng lưu trữ dữ liệu lớn, nó chuyên quản lý tài nguyên (chủ yếu là cơ sở dữ liệu), cung cấp các dịch vụ mạng cho các máy khách (client) sử dụng. Bình thường nó chạy suốt thời gian thực và sẵn sàng chấp nhận các yêu cầu kết nối và các yêu cầu dịch vụ khác từ máy khách.

### Máy trạm (Client)

Client là máy khách, nó thường được sử dụng bởi người dùng cuối. Nó hoạt động dựa trên việc sử dụng dịch vụ mà máy server cung cấp để thực hiện các công việc mà người dùng cuối mong muốn.

### Quy trình hoạt động

Quy trình hoạt động của mô hình này lặp lại 2 quá trình như sau:

* Client gửi yêu cầu lên server.
* Server nhận được yêu cầu thì sẽ xử lý thích hợp và trả lời lại client.

A close-up of a web browser

Description automatically generated

## Giới thiệu giao thức *TCP/IP*

### Giới thiệu về TCP

Giao thức TCP (Transmission Control Protocol) là giao thức hướng kết nối (connection-oriented), nó đòi hỏi thiết lập kết nối trước khi bắt đầu gửi dữ liệu và kết thúc kết nối khi việc gửi dữ liệu hoàn tất theo đúng thứ tự: thiết lập kết nối, truyền dữ liệu và kết thúc kết nối.

TCP xác định cách các ứng dụng tạo kênh giao tiếp trong mạng. Ngoài ra, nó cũng quản lý cách các tin được phân thành các gói nhỏ trước khi được chuyển qua Internet và được tập hợp lại theo đúng thứ tự tại địa chỉ đến.

### Giới thiệu về IP

IP là địa chỉ của một máy tính trên mạng, dựa vào địa chỉ IP giao thức TCP có thể truyền dữ liệu chính xác từ một máy này qua máy kia thông qua hệ thống mạng. Ở trên mạng, một máy tính sẻ có một địa chỉ IP khác nhau, từ địa chỉ IP có thể biết được máy nào trên mạng và ngược lại.

### Giới thiệu về cổng (Port)

Port là giao thức bit 16 đứng đầu (chèn vào phần đầu header) của mỗi gói tin trong giao thức TCP, UDP hay còn gọi là cổng port, nơi quy định các tập dữ liệu riêng biệt. Port là một dạng thuật toán đã được định sẵn và mỗi máy tính cần phải có thì mới có thể nhận và gửi các gói tin đi được. Khi hoạt động Port cũng được quy đổi giống với số bit của bất kì một mã dữ liệu nào đó. Đơn giản thì đây giống như cánh cổng có quyền cho vào hay không với các dữ liệu muốn ra vào hệ thống máy tính.

Cần 1 địa chỉ IP, bạn xác định được một máy trong một mạng, nhưng nếu chúng chạy cùng nhiều dịch vụ khác nhau thì phải có cách để phân biệt được chúng và lúc này cần đến Port.

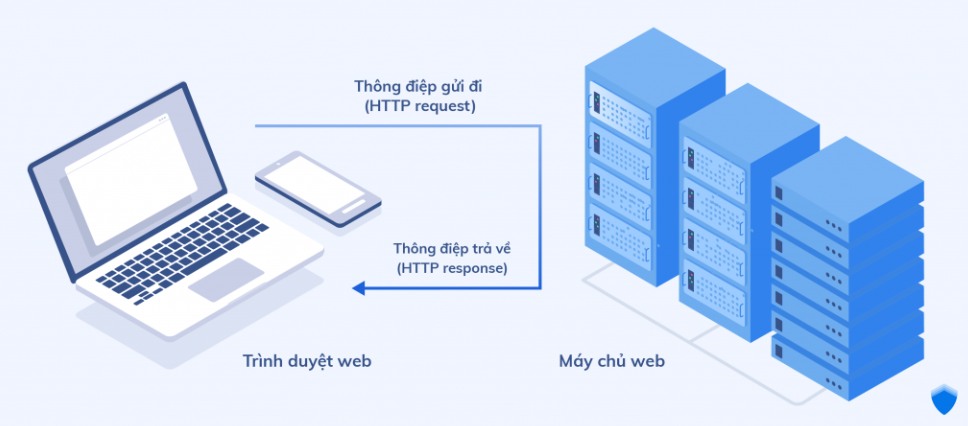
## Giới thiệu giao thức Http/Https

### Giao thức Http

HTTP (Hypertext Transfer Protocol) là giao thức truyền tải siêu văn bản. Đây là giao thức tiêu chuẩn cho World Wide Web (www) để truyền tải dữ liệu dưới dạng văn bản, âm thanh, hình ảnh, video từ Web Server tới trình duyệt web của người dùng và ngược lại.

HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet). Bộ giao thức TCP/IP là một bộ các giao thức truyền thông cài đặt chồng giao thức mà Internet và hầu hết các mạng máy tính thương mại đang chạy trên đó. Bộ giao thức này được đặt theo tên hai giao thức chính là TCP (Transmission Control Protocol – Giao thức điều khiển truyền vận) và IP (Internet Protocol – Giao thức Internet).

HTTP hoạt động theo mô hình Client (máy khách) – Server (máy chủ). Việc truy cập website được tiến hành dựa trên các giao tiếp giữa 2 đối tượng trên. Khi bạn truy cập một trang web qua giao thức HTTP, trình duyệt sẽ thực hiện các phiên kết nối đến server của trang web đó thông qua địa chỉ IP do hệ thống phân giải tên miền DNS cung cấp. Máy chủ sau khi nhận lệnh, sẽ trả về lệnh tương ứng giúp hiển thị website, bao gồm các nội dung như: văn bản, ảnh, video, âm thanh,…



### Giao thức Https

HTTPS hoạt động tương tự như HTTP, tuy nhiên được bổ sung thêm chứng chỉ SSL (Secure Sockets Layer – tầng ổ bảo mật) hoặc TLS (Transport Layer Security – bảo mật tầng truyền tải). Hiện tại, đây là các tiêu chuẩn bảo mật hàng đầu cho hàng triệu website trên toàn thế giới

HTTPS hoạt động theo mô hình Client (máy khách) – Server (máy chủ). Việc truy cập website được tiến hành dựa trên các giao tiếp giữa 2 đối tượng trên. Khi bạn truy cập một trang web qua giao thức HTTP, trình duyệt sẽ thực hiện các phiên kết nối đến server của trang web đó thông qua địa chỉ IP do hệ thống phân.

## Giới thiệu về các công nghệ sử dụng trong đề tài

### Giới thiệu về ReactJS

ReactJS (hoặc React) là một thư viện JavaScript phát triển bởi Facebook, được sử dụng để xây dựng giao diện người dùng (UI) cho ứng dụng web. React được ra đời vào năm 2013 và đã trở thành một trong những công cụ phổ biến nhất để phát triển ứng dụng web hiện đại.

* ***Đặt tính của React JS*:**

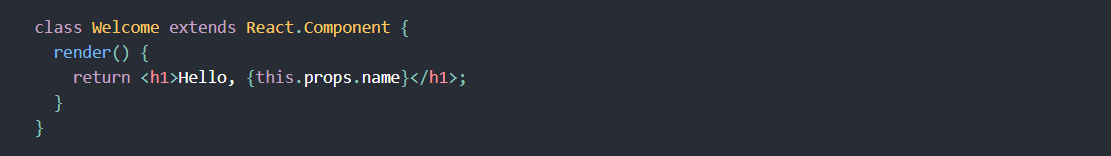
Tư tưởng React JS là xây dựng lên các components có tính tái sử dụng , dễ dàng cho việc chia nhỏ vấn đề , testing . Nó giúp chúng ta dễ dàng quản lý , mở rộng hệ thống , điều này nếu là Angular JS thì đòi hỏi cấu trúc , cách viết code phải thật tối ưu .

React JS luôn giữ các components ở trạng thái stateless ( nhiều nhất có thể ) khiến ta dễ dàng quản lý bởi nó chẳng khác gì một trang static HTML . Bản thân các components này không có trạng thái ( state ) nó nhận đầu vào từ bên ngoài và chỉ hiện thị ra dựa vào các đầu vào đó , điều này lý giải tại sao nó lại mang tính tái sử dụng ( reuse ) và dễ dàng cho việc.

* ***Thư viện Component-Based:***

React được thiết kế theo mô hình component-based, cho phép bạn xây dựng giao diện bằng cách tạo và tái sử dụng các thành phần (components) độc lập. Điều này giúp quản lý mã nguồn dễ dàng, tăng tính tái sử dụng và giảm phức tạp của ứng dụng.

Mỗi component trong React có một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái. Mọi thứ React đều là component. Chúng giúp bảo trì mã code khi làm việc với các dự án lớn. Một react component đơn giản chỉ cần một method render. Có rất nhiều methods khả dụng khác, nhưng render là method chủ đạo.



Tạo một component

* ***Props và State:***

Props: giúp các component tương tác với nhau, component nhận input gọi là props, và trả thuộc tính mô tả những gì component con sẽ render. Prop là bất biến. State: thể hiện trạng thái của ứng dụng, khi state thay đồi thì component đồng thời render lại để cập nhật UI.

* ***Hỗ trợ JSX:***

Trong React, thay vì thường xuyên sử dụng JavaScript để thiết kế bố cục trang web thì sẽ dùng JSX. JSX được đánh giá là sử dụng đơn giản hơn JavaScript và cho phép trích dẫn HTML cũng như việc sử dụng các cú pháp thẻ HTML để render các subcomponent. JSX tối ưu hóa code khi biên soạn, vì vậy nó chạy nhanh hơn so với code JavaScript tương đương.

* ***Virtual DOM:***

React sử dụng một cơ chế gọi là Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Thay vì cập nhật trực tiếp DOM mỗi khi dữ liệu thay đổi, React tạo ra một bản sao ảo của DOM (Virtual DOM) và so sánh nó với DOM thật để cập nhật chỉ những phần thay đổi cần thiết. Điều này giúp giảm tải và làm cho ứng dụng chạy nhanh hơn.

*Thêm vào đó, React.js có khả năng gửi các loại yêu cầu HTTP như GET, POST, PUT, DELETE và nhiều loại khác thông qua thư viện hoặc gói HTTP như Axios hoặc Fetch API. Gói HTTP như Axios hoặc Fetch API cho phép bạn tạo và gửi yêu cầu HTTP từ phía máy khách (frontend) đến phía máy chủ (backend) và xử lý các phản hồi từ máy chủ.*

* ***Axios****:*

Axios là một thư viện JavaScript phổ biến được sử dụng trong ứng dụng React để tạo và quản lý các yêu cầu HTTP.

Axios cung cấp một API dễ sử dụng cho việc gửi các loại yêu cầu HTTP khác nhau và xử lý phản hồi và lỗi

* ***Fetch*:**

fetch() là một hàm của đối tượng window trong javascript, cho phép chúng ta lấy dữ liệu từ API một cách bất đồng bộ mà không cần cài đặt thêm thư viện, nghĩa là fetch không phải là một thư viện mà là công cụ có sẵn mà trình duyệt cung cấp thông qua Fetch API

*Cả Axios và Fetch API đều có thể được sử dụng để gửi yêu cầu HTTP trong ứng dụng React, và lựa chọn giữa chúng thường phụ thuộc vào sở thích cá nhân và yêu cầu cụ thể của dự án.*

### Giới thiệu về NodeJS

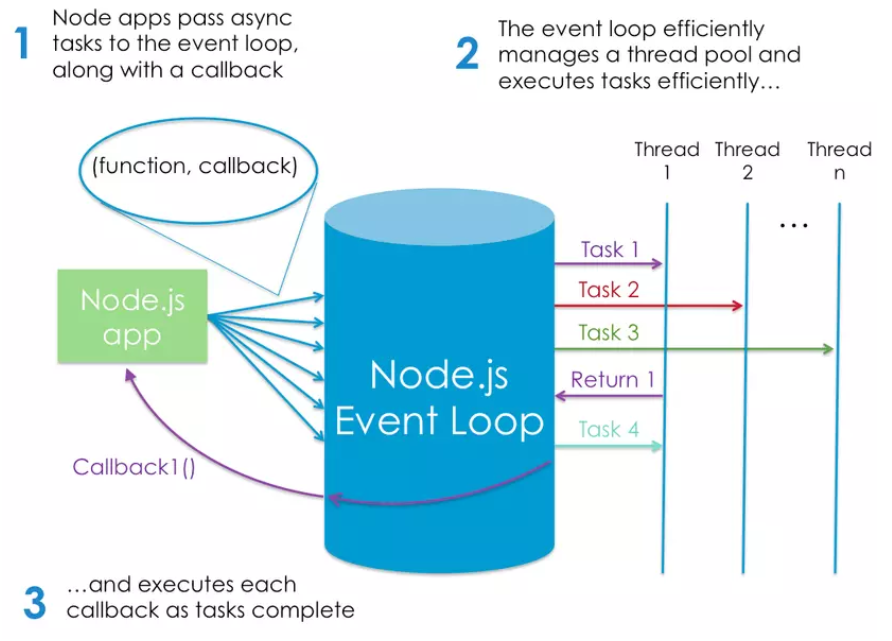
Node.js là một môi trường thực thi mã JavaScript phía máy chủ (server-side) được xây dựng dựa trên engine JavaScript V8 của Google. Nó cho phép bạn chạy mã JavaScript trên máy chủ thay vì trình duyệt web, mở rộng khả năng sử dụng ngôn ngữ này từ phía client-side (trình duyệt) sang server-side (máy chủ).

* ***Bất đồng bộ:***

Node.js làm việc chủ yếu dựa trên mô hình bất đồng bộ, cho phép xử lý đồng thời nhiều yêu cầu I/O (như đọc/ghi dữ liệu từ cơ sở dữ liệu, tệp tin, giao tiếp mạng) mà không cần chờ đợi hoàn thành của mỗi yêu cầu trước khi tiếp tục xử lý các yêu cầu khác. Điều này giúp cải thiện hiệu suất và đáp ứng của ứng dụng.

* ***Singled- Threaded:***

Mô hình hoạt động của Node.js là một mô hình đơn luồng, không đồng bộ (single-threaded, non-blocking) và sự kiện-driven (event-driven). Điều này có nghĩa rằng Node.js sử dụng một luồng duy nhất để xử lý nhiều yêu cầu I/O và sự kiện mà không phải chờ đợi cho việc hoàn thành của mỗi yêu cầu trước khi xử lý yêu cầu tiếp theo.



. Mô hình hoạt động NodeJS

* ***Sự kiện và Bộ lập trình sự kiện (Event Loop):***

Node.js sử dụng mô hình bộ lập trình sự kiện (Event Emitters) để quản lý các sự kiện và xử lý sự kiện. Bất kỳ hoạt động I/O nào như đọc/ghi dữ liệu từ tệp tin, truy vấn cơ sở dữ liệu, hoặc gửi yêu cầu HTTP cũng được thực hiện bằng cách đăng ký các hàm xử lý sự kiện (callbacks). Khi sự kiện xảy ra, Node.js gọi các hàm xử lý sự kiện này.

* ***Non-Blocking I/O:***

Khi một yêu cầu I/O được gửi đi, Node.js không chờ đợi kết quả trả về. Thay vào đó, nó tiếp tục xử lý các yêu cầu khác trong bộ lập trình sự kiện. Khi kết quả I/O sẵn sàng, Node.js gọi callback tương ứng để xử lý dữ liệu.

* ***Bộ luồng (Thread Pool):***

Node.js sử dụng một bộ lập trình sự kiện và một bộ luồng riêng để xử lý các yêu cầu I/O chậm (như đọc tệp tin lớn hoặc kết nối mạng). Bộ luồng này cho phép xử lý các yêu cầu I/O mà không ảnh hưởng đến luồng chính và không làm chậm sự kiện chính.

* ***Cơ sở dữ liệu và Mạng:***

Node.js cung cấp các thư viện cho việc giao tiếp với cơ sở dữ liệu (SQL hoặc NoSQL) và làm việc với giao thức mạng như HTTP và WebSocket.

Một điểm mạnh nữa của Node đó là nó cực thích hợp với JSON. JSON được coi là lý tưởng khi được sử dụng bởi một chương trình JavaScript. Khi làm việc với Node, dữ liệu có thể chuyển qua các tầng layer mà không cần phải reformat.

Cuối cùng, JavaScript thực sự quá phổ biến, điều này giúp cho việc chuyển sang phát triển Node app dễ dàng hơn so với những ngôn ngữ phía server khác.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## Phân tích chức năng của chương trình

Đề tài của nhóm là “Xây dựng chương trình Client/Server để in lịch thế kỷ” được chia thành 2 thành phần chính là phía Máy Chủ (Server) và phía Máy Khách (Client)

Phát triển ứng dụng theo kiến trúc Client/Server: Chương trình dựa trên mô hình kiến trúc client/server, tập trung vào việc triển khai trên nền mạng đa máy (ví dụ, mạng cục bộ...). Để đáp ứng yêu cầu này, chúng ta sẽ tuân thủ phương pháp đã được giới thiệu trong các phần trước, dựa trên các cơ sở lý thuyết đã được trình bày.

Tiếp nhận yêu cầu xem lịch của năm, tháng cụ thể và trả về thông tin lịch cần tìm: Đầu vào là tháng, năm có định dạng số, và đầu ra là thông tin lịch dưới dạng text/html trả về cho client.

### Máy chủ (Server):

* ***Tiếp nhận yêu cầu :***

Máy chủ lắng nghe yêu cầu và sử dụng phần back-end để xử lý yêu cầu.

* ***Tính toán lịch cho năm đã nhập:***

Máy chủ sử dụng thuật toán để tính toán lịch cho tháng, năm đã được cung cấp từ máy khách.

* ***Trả về kết quả về phía máy khách:***

Máy chủ sau đó gửi kết quả (lịch cho năm đã nhập) trở lại máy khách thông qua phản hồi HTTP.

* ***Bảo mật và kiểm soát lỗi:***

Chương trình được thiết kế để đảm bảo tính bảo mật của dữ liệu được truyền tải giữa máy khách và máy chủ. Nếu có lỗi xảy ra trong quá trình xử lý yêu cầu hoặc tính toán lịch, chương trình phải xử lý lỗi đó một cách an toàn.

### Máy trạm (Client):

* ***Nhập yêu cầu từ phía máy khách (client):***

Cho phép người dùng nhập vào một năm hoặc tháng từ giao diện người dùng trên máy khách.

* ***Gửi yêu cầu từ máy khách đến máy chủ:***

Sau khi người dùng nhập năm, máy khách sử dụng gói HTTP (như Axios hoặc Fetch API) để gửi yêu cầu HTTP POST chứa năm đó đến máy chủ.

* ***Hiển thị lịch trên giao diện người dùng của máy khách:***

Máy khách nhận kết quả từ máy chủ và hiển thị lịch tương ứng lên giao diện người dùng, để người dùng có thể xem lịch cho năm đã nhập.

## Phương pháp xử lí

# TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## Các công cụ sử dụng

## Kết quả

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

[1] Tên tác giả, Tên tài liệu, Tên nhà xuất bản, năm xuất bản

[2] Tên chủ sở hữu, Tên bài viết, url, ngày truy cập

PHỤ LỤC