

TRƯỜNG ĐẠI HỌC KIÊN GIANG
KHOA THÔNG TIN & TRUYỀN THÔNG



BÁO CÁO THỰC TẬP

Đề tài:
**XÂY DỰNG ỨNG DỤNG WEB
QUẢN LÝ CÔNG VIỆC**

Sinh viên: Nguyễn Nhật Đăng

MSSV: 21072006005

Lớp: B021TT1

GVHD: ThS. Võ Hoàng Nhân

Tháng 5 năm 2024

LỜI CẢM ƠN

Xin gửi lời tri ân sâu sắc đến Thầy ThS. VÕ HOÀNG NHÂN - giảng viên hướng dẫn thực tập đã tận tình dù dắt, hỗ trợ em trong suốt quá trình thực hiện đề tài “Xây dựng ứng dụng web quản lý công việc”. Sự nhiệt tình, tâm huyết và những lời khuyên quý giá của Thầy/Cô đã giúp em hoàn thành đề tài một cách tốt nhất. Em cũng xin chân thành cảm ơn Ban Giám đốc cùng toàn thể cán bộ, nhân viên công ty FPT Telecom Rạch Giá đã tạo điều kiện cho em thực tập tại công ty, được trải nghiệm môi trường làm việc thực tế và học hỏi những kiến thức, kỹ năng quý báu. Báo cáo thực tập này là kết quả của sự nỗ lực, cố gắng trong suốt thời gian thực tập của em. Tuy nhiên, do thời gian thực tập có hạn, kinh nghiệm và kiến thức còn nhiều hạn chế, nên báo cáo có thể còn thiếu sót hoặc chưa đầy đủ nội dung theo yêu cầu. Vì vậy, sự động viên đóng góp ý kiến của các thầy cô, các bạn sinh viên trong lớp trong khoa sẽ là những ý kiến hết sức quý báu giúp đề tài được hoàn thiện hơn. Em xin chân thành cảm ơn!

Kiên Giang, ngày.....tháng.....năm 202

Sinh viên thực hiện

NHẬN XÉT CỦA NOI THỰC TẬP

Tổng điểm:

....., ngày.... tháng.... năm.....

NGƯỜI ĐẠI DIỆN

(ký tên và đóng dấu)

NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN

Tổng điểm:

....., ngày.... tháng.... năm.....

NGƯỜI HƯỚNG DẪN

(ký tên và ghi họ tên)

GIỚI THIỆU VỀ NƠI THỰC TẬP: CÔNG TY VIỄN THÔNG FPT TELECOM RẠCH GIÁ



Tên cơ quan thực tập: Công ty viễn thông FPT Telecom Rạch Giá.

Địa chỉ: 259 Nguyễn Bình Khiêm, Vĩnh Thanh, Rạch Giá, Kiên Giang.

Chức năng: Cung cấp các dịch vụ lắp đặt, sửa chữa internet và internet truyền hình.

NHẬT KÝ THỰC TẬP
Từ ngày 18/03/2024 đến ngày 28/04/2024

Thời gian	Nội dung công việc	Nhận xét của giảng viên về thái độ và tiến độ	Các vấn đề cần tiến hành chỉnh sửa	Giảng viên đánh giá nội dung đã chỉnh sửa	Chữ ký của GVHD	Chữ ký của sinh viên thực tập
Tuần 1						
Thứ 2	- Làm quen với môi trường làm việc - Nhận người hướng dẫn					
Thứ 3	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 4	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 5	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 6	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Tuần 2						
Thứ 2	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 3	- Cài đặt module wifi cho khách hàng					
Thứ 4	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 5	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 6	- Nghỉ phép					

Tuần 3						
Thứ 2	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 3	- Cài đặt module wifi cho khách hàng - Kéo dây mạng					
Thứ 4	- Cài đặt module wifi cho khách hàng - Kéo dây mạng					
Thứ 5	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 6	- Nghỉ phép					
Tuần 4						
Thứ 2	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 3	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 4	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 5	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 6	- Nghỉ phép					
Tuần 5						
Thứ 2	- Đi với người hướng dẫn, đến nhà khách hàng làm việc					
Thứ 3	- Cài đặt module wifi cho khách hàng - Kéo dây mạng					
Thứ 4	- Cài đặt module wifi cho khách hàng - Kéo dây mạng					

Thứ 5	- Cài đặt module wifi cho khách hàng - Kéo dây mạng					
Thứ 6	- Nghỉ phép					
Tuần 6						
Thứ 2	- Đi với người hướng dẫn, đến nhà khách hang làm việc					
Thứ 3	- Đi với người hướng dẫn, đến nhà khách hang làm việc					
Thứ 4	- Đi với người hướng dẫn, đến nhà khách hang làm việc					
Thứ 5	- Đi với người hướng dẫn, đến nhà khách hang làm việc					
Thứ 6	- Nghỉ phép					

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU CHUNG	1
1.1. Giới thiệu tổng quan	1
1.2. Giới thiệu đè tài	2
1.2.1. Chức năng tạo tài khoản	2
1.2.2. Chức năng đăng nhập	2
1.2.3. Quản lý các công việc	3
1.2.4. Quản lý các danh sách công việc	3
1.2.5. Chia sẻ các danh sách công việc cho người dùng khác	3
1.3. Giới thiệu về cơ sở lý thuyết cho đè tài	3
1.3.1. Mô hình client-server	3
1.3.2. HTTP protocol	4
1.3.3. Công nghệ được sử dụng	4
CHƯƠNG 2: THIẾT KẾ CÁC MÔ HÌNH HỆ THỐNG	9
2.1. Mô hình quan niệm	9
2.1. Mô hình luận lý	9
2.3. Mô hình Vật lý	9
2.4. Mô hình liên kết các bảng trên máy tính	12
2.5. Mô hình xử lý (UML)	13
2.5.1. Mô hình use case	13
CHƯƠNG 3: CÁC MỤC XỬ LÝ (DEMO)	14
3.1. Trang chủ	14
3.2. Đăng nhập hệ thống	14
3.2. Tạo tài khoản	15

3.2. Cập nhật: thêm, sửa, xóa	17
3.2.1. Cập nhật list	17
3.2.2. Cập nhật công việc	19
3.3. Tìm kiếm, hiển thị, xử lý thông tin theo yêu cầu đề tài.....	22
3.4. Thống kê, tổng hợp số liệu theo yêu cầu đề tài	24
3.5. Chia sẻ danh sách	25
CHƯƠNG 4: KẾT LUẬN VÀ KIẾN NGHỊ	28
4.2. Hướng phát triển đề tài	28
4.3. Kiến nghị	29

DANH MỤC HÌNH ẢNH

Hình 1: Mô hình quan niệm.....	9
Hình 2: Mô hình liên kết các bảng trên máy tính	12
Hình 3: Mô hình usecase	13
Hình 4: Giao diện trang chủ	14
Hình 5: Giao diện ứng dụng	14
Hình 6: Giao diện đăng nhập	15
Hình 7: Giao diện tạo tài khoản.....	16
Hình 8: Giao diện thêm danh sách	17
Hình 9: Giao diện sửa danh sách	18
Hình 10: Giao diện xóa danh sách.....	18
Hình 11: Giao diện thêm công việc	19
Hình 12: Giao diện công việc	21
Hình 13: Giao diện tìm kiếm	23
Hình 14: Giao diện kết quả tìm kiếm	23
Hình 15: Giao diện xem danh sách.....	24
Hình 16: Giao diện chia sẻ danh sách	25

DANH MỤC BẢNG

Bảng 1: Mô hình vật lý User	10
Bảng 2: Mô hình vật lý List.....	10
Bảng 3: Mô hình vật lý Share.....	10
Bảng 4: Mô hình vật lý Task	11
Bảng 5: Mô hình vật lý Subtask	11
Bảng 6: Mô hình vật lý Comment	11

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1. Giới thiệu tổng quan

Trong xã hội hiện đại, con người ngày càng bận rộn với nhiều công việc khác nhau, dẫn đến tình trạng hay quên, lỡ hẹn hoặc không quản lý công việc hiệu quả. Việc sử dụng các ứng dụng quản lý công việc ngày càng phổ biến để giúp mọi người sắp xếp, theo dõi và hoàn thành công việc một cách khoa học và hiệu quả hơn.

Nhu cầu sử dụng ứng dụng quản lý công việc ngày càng tăng cao, tuy nhiên trên thị trường hiện nay có nhiều ứng dụng với tính năng phức tạp, giao diện rườm rà, khó sử dụng. Do vậy, việc xây dựng một ứng dụng web quản lý công việc với giao diện đơn giản, dễ sử dụng, tính năng đa dạng và hiệu quả là rất cần thiết.

Theo báo cáo của Statista, thị trường phần mềm quản lý công việc toàn cầu dự kiến đạt giá trị 18,13 tỷ USD vào năm 2027, với tốc độ tăng trưởng kép hàng năm (CAGR) là 16,7% trong giai đoạn từ năm 2022 đến năm 2027.

Nhu cầu theo đối tượng người dùng:

- Học sinh, sinh viên: Sử dụng để quản lý bài tập, lịch học và các hoạt động học tập khác.
- Nhân viên văn phòng: Sử dụng để quản lý công việc, dự án và deadline.
- Freelancer: Sử dụng để quản lý các dự án, khách hàng và thời gian làm việc.
- Các doanh nghiệp: Sử dụng để quản lý quy trình làm việc, phân công nhiệm vụ và theo dõi tiến độ công việc.

Phạm vi đề tài:

- Ứng dụng web quản lý công việc sẽ được xây dựng trên nền tảng web với các ngôn ngữ lập trình phổ biến như HTML, CSS, Javascript và framework lập trình web phù hợp.
- Ứng dụng sẽ bao gồm các tính năng cơ bản như: tạo danh sách công việc, sắp xếp theo thứ tự ưu tiên, đặt thời hạn, theo dõi tiến độ.
- Có thể mở rộng thêm các tính năng nâng cao như: chia sẻ danh sách công việc với người khác, tạo nhóm làm việc, quản lý dự án, v.v.

Khả năng ứng dụng triển khai đề tài:

- Ứng dụng web quản lý công việc có thể được áp dụng cho nhiều đối tượng người dùng khác nhau như: học sinh, sinh viên, nhân viên văn phòng, freelancer, v.v.
- Ứng dụng có thể được sử dụng trên nhiều thiết bị khác nhau như máy tính, điện thoại thông minh, máy tính bảng.

Việc triển khai ứng dụng web quản lý công việc có thể mang lại nhiều lợi ích như:

- Nâng cao hiệu quả quản lý công việc.
- Tiết kiệm thời gian và chi phí.
- Tăng năng suất làm việc.
- Giảm thiểu sai sót và rủi ro.

1.2. Giới thiệu đề tài

Đề tài tập trung vào việc phát triển một ứng dụng web quản lý công việc với các chức năng chính như:

- Chức năng tạo tài khoản.
- Chức năng đăng nhập.
- Quản lý các công việc.
- Quản lý các danh sách công việc.
- Chia sẻ các danh sách công việc cho người dùng khác.

1.2.1. Chức năng tạo tài khoản

Người dùng muốn sử dụng ứng dụng phải tạo một tài khoản cá nhân người dùng phải cung cấp các thông tin như email (người dùng phải xác thực email của mình bằng otp để có thể đăng ký tài khoản) tên người dùng (tên người dùng là duy nhất và không trùng lặp giữa các người dùng với nhau) và mật khẩu (các kí tự chữ cái và số có độ dài từ 8-30 kí tự).

1.2.2. Chức năng đăng nhập.

Người dùng phải đăng nhập mới có thể sử dụng ứng dụng. Người dùng phải nhập tài khoản và mật khẩu được đăng ký trước đó, nếu không có tài khoản người dùng phải tạo một tài khoản mới.

1.2.3. Quản lý các công việc

Người dùng có thể tạo các công việc cần làm, mỗi công việc chưa các thông tin như tên công việc, mô tả, thời gian đến hạn của công việc, công việc có lặp lại hay không (tự động tạo các công việc lặp đi lặp lại cho người dùng khi họ hoàn thành), độ quan trọng của công việc, các công việc phụ, và các bình luận về công việc.

1.2.4. Quản lý các danh sách công việc

Mỗi người dùng có thể tạo mới sửa và xóa các danh sách công việc. Danh sách công việc chứa các công việc cần thực hiện, người dùng có thể sắp xếp chúng. Việc có thể tạo nhiều danh sách công việc giúp người dùng quản lý công việc dễ dàng hơn, bằng cách nhóm các công việc lại với nhau thành từng danh sách cụ thể.

1.2.5. Chia sẻ các danh sách công việc cho người dùng khác

Người dùng có thể chia sẻ các danh sách công việc cho người dùng khác. Người dùng được chia sẻ có quyền truy cập vào danh sách công việc, thêm sửa xóa các công việc. Chủ sở hữu danh sách có thể tạo một liên kết để chia sẻ danh sách của họ và có thể vô hiệu hóa liên kết để ngưng việc chia sẻ. Bất kỳ ai có liên kết đó sẽ có quyền truy cập vào danh sách, chủ sở hữu có quyền quản lý các người truy cập vào danh sách của mình.

1.3. Giới thiệu về cơ sở lý thuyết cho đề tài

Các cơ sở lý thuyết để xây dựng phần mềm:

- Cơ sở dữ liệu.
- Phân tích thiết kế hệ thống thông tin.
- Lập trình web.

1.3.1. Mô hình client-server.

Mô hình client-server(máy khách-máy chủ) là một mô hình kiến trúc mạng trong đó các máy tính được chia thành hai nhóm: máy khách và máy chủ. Máy khách là các máy tính yêu cầu dịch vụ từ máy chủ. Máy chủ là các máy tính cung cấp dịch vụ cho máy khách.

Mô hình máy khách-máy chủ là một mô hình rất linh hoạt và có thể mở rộng có thể được sử dụng cho nhiều ứng dụng khác nhau.

1.3.2. HTTP protocol

Hypertext Transfer Protocol (HTTP) là giao thức lớp ứng dụng để truyền các tài liệu siêu phong tiện, chẳng hạn như HTML. Nó được thiết kế để liên lạc giữa trình duyệt web và máy chủ web, nhưng nó cũng có thể được sử dụng cho các mục đích khác. HTTP tuân theo mô hình máy khách-máy chủ cổ điển. Mỗi yêu cầu HTTP được gửi từ trình duyệt web tới máy chủ thông qua một kết nối mới, và sau khi máy chủ phản hồi, kết nối sẽ được đóng. Điều này có nghĩa là trong mô hình HTTP truyền thống, mỗi yêu cầu chỉ được thực hiện theo kiểu "gửi yêu cầu - nhận phản hồi" và sau đó kết thúc. Ưu điểm của HTTP:

- Đơn giản và dễ sử dụng: Dễ dàng triển khai và học hỏi.
- Linh hoạt: Hỗ trợ nhiều phương thức và mã trạng thái.
- Phổ biến: Hỗ trợ bởi tất cả các trình duyệt web và máy chủ web. Nhược điểm của HTTP:
 - Không thời gian thực: Dữ liệu có thể bị trì hoãn.
- Không hiệu quả: Sử dụng nhiều tài nguyên hơn so với WebSocket cho các ứng dụng thời gian thực.

1.3.3. Công nghệ được sử dụng

1.3.3.1. Ngôn ngữ lập trình

1. JavaScript

JavaScript là một ngôn ngữ lập trình phía máy khách (client-side) và phía máy chủ (server-side) được sử dụng rộng rãi trong việc phát triển ứng dụng web. Ban đầu JavaScript được tạo ra để tạo ra các tương tác động trên trang web, nhưng ngày nay nó đã phát triển thành một ngôn ngữ lập trình đa nền tảng mạnh mẽ.

Các đặc điểm của JavaScript bao gồm:

- Kịch bản (Scripting): JavaScript thường được sử dụng để viết các đoạn mã ngắn và tương tác trực tiếp với các phần tử HTML trên trang web.
- Đa nền tảng: JavaScript chạy trên mọi trình duyệt web phổ biến và cũng có thể được sử dụng với các công cụ phía máy chủ như Node.js để phát triển ứng dụng máy chủ.
- Động (Dynamic): JavaScript là một ngôn ngữ động, cho phép thay đổi cấu trúc và kiểu dữ liệu của biến trong quá trình thực thi.

- Cú pháp dễ hiểu: JavaScript có một cú pháp dễ hiểu và linh hoạt, giúp việc viết mã trở nên nhanh chóng và dễ dàng hơn.

JavaScript được sử dụng rộng rãi trong việc xây dựng các ứng dụng web đa dạng, bao gồm các trò chơi trực tuyến, ứng dụng di động, ứng dụng web thời gian thực, và nhiều hơn nữa.

2. TypeScript

TypeScript là một ngôn ngữ lập trình mã nguồn mở, phát triển bởi Microsoft, là một phiên bản mở rộng của JavaScript. TypeScript giúp mở rộng khả năng của JavaScript bằng cách thêm tính năng kiểu tĩnh và các tính năng nâng cao khác, nhằm tăng tính bảo mật, khả năng duy trì và hiệu suất của mã.

Các đặc điểm của JavaScript bao gồm:

- Kiểu tĩnh: TypeScript hỗ trợ kiểu dữ liệu tĩnh, giúp phát hiện lỗi trong quá trình phát triển và cung cấp gợi ý thông qua IDE.
- Mở rộng JavaScript: TypeScript là một phiên bản mở rộng của JavaScript, vì vậy mã JavaScript hiện có có thể được sử dụng trong dự án TypeScript mà không cần thay đổi.
- Tính năng hiện đại: TypeScript hỗ trợ các tính năng hiện đại của ECMAScript (ES), bao gồm lớp, module, hàm mũi tên và nhiều hơn nữa.
- Công cụ hỗ trợ: TypeScript được hỗ trợ bởi các trình biên dịch và IDE phổ biến, cung cấp khả năng kiểm tra lỗi tĩnh và phát hiện lỗi trong quá trình phát triển.

TypeScript được sử dụng rộng rãi trong các dự án lớn và phức tạp, đặc biệt là trong phát triển ứng dụng web và ứng dụng di động. Nó giúp tăng tính bảo trì, khả năng mở rộng và hiệu suất của mã nguồn, đồng thời cung cấp một quy trình phát triển rõ ràng và an toàn hơn.

3. HTML

HTML viết tắt của Hypertext Markup Language là script dùng để xây dựng và cấu trúc lại các thành phần có trong Website.

HTML tạm dịch là script đánh dấu siêu văn bản. Người ta thường sử dụng HTML trong việc phân chia các đoạn văn, heading, links, blockquotes, ... giúp câu thành các cấu trúc cơ bản của một Website, làm cho trang Web trở thành một hệ thống hoàn chỉnh. Cụ

thể, ngôn ngữ đánh dấu siêu văn bản này giúp bố cục, chia khung sườn các thành phần trang Web. Đồng thời, nó còn hỗ trợ khai báo các File kỹ thuật số như nhạc, Video, hình ảnh,

4. CSS

CSS là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web. Chúng ta có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng... thì CSS sẽ giúp chúng ta có thể thêm style vào các phần tử HTML đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ, thay đổi cấu trúc... CSS được phát triển bởi W3C (World Wide Web Consortium) vào năm 1996, vì HTML không được thiết kế để gắn tag để giúp định dạng trang web. Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

1.3.3.2. Library và Framework

Phần mềm sử dụng nhiều library và framework đây là các phần phụ thuộc chính của ứng dụng

1. Next.js

Next.js là một framework phát triển ứng dụng web phía máy khách và phía máy chủ được xây dựng trên nền tảng JavaScript. Nó được xây dựng trên cơ sở của React, một thư viện UI phổ biến, và cung cấp các tính năng mạnh mẽ để phát triển ứng dụng web độc lập và tối ưu hóa trải nghiệm người dùng.

2. Honojs

Hono - [炎] mang ý nghĩa "flame" hay "lửa" trong tiếng Nhật - là một framework web nhỏ gọn, đơn giản và siêu nhanh dành cho các Edge. Nó hoạt động trên bất kỳ động cơ JavaScript nào: Cloudflare Workers, Fastly Compute, Deno, Bun, Vercel, Netlify, AWS Lambda, Lambda@Edge và Node.js.

3. Prisma

Prisma là một công cụ ORM (Object-Relational Mapping) hiệu suất cao và đa nền tảng dành cho việc làm việc với cơ sở dữ liệu trong ứng dụng web. Nó cung cấp một cách tiện lợi để tương tác với cơ sở dữ liệu bằng cách sử dụng ngôn ngữ truy vấn riêng gọi là Prisma Query Language (PQL). Prisma có thể định nghĩa mô hình dữ liệu của ứng dụng bằng cách sử dụng ngôn ngữ Schema Definition Language (SDL) dễ đọc và dễ hiểu. Prisma sẽ sử dụng mô hình này để tạo ra các bảng trong cơ sở dữ liệu và cung cấp các phương thức truy vấn và tạo/sửa/xóa dữ liệu cho các bảng. Prisma hỗ trợ nhiều loại cơ sở dữ liệu phổ biến như PostgreSQL, MySQL và SQLite.

4. Tailwindcss

Tailwind CSS là một framework CSS hướng utility, giúp xây dựng giao diện web tùy chỉnh nhanh chóng và dễ dàng. Điểm đặc biệt của Tailwind CSS là nó không áp đặt bất kỳ kiểu thiết kế nào mặc định, mà thay vào đó cung cấp một bộ các class utility linh hoạt, cho phép xây dựng giao diện theo ý muốn của riêng mình.

1.3.3.3. Database

1. PostgreSQL

PostgreSQL (hay còn gọi tắt là **Postgres**) là hệ thống quản trị cơ sở dữ liệu quan hệ và đối tượng (**ORDBMS**) miễn phí và mã nguồn mở tiên tiến nhất hiện nay. Nó được phát triển dựa trên POSTGRES 4.2 tại phòng khoa học máy tính Berkeley, Đại học California.

PostgreSQL được đánh giá cao bởi:

- Hiệu suất cao: Xử lý truy vấn nhanh chóng, đáp ứng tốt nhu cầu cho ứng dụng có lượng truy cập lớn.
- Độ tin cậy cao: Ôn định, ít xảy ra lỗi, đảm bảo an toàn cho dữ liệu.
- Khả năng mở rộng: Dễ dàng mở rộng dung lượng lưu trữ khi cần thiết.
- Tính linh hoạt: Hỗ trợ nhiều tính năng, giao diện lập trình (API) và tích hợp tốt với ứng dụng web/di động.
- Mã nguồn mở: Miễn phí, cho phép tùy chỉnh và phát triển theo nhu cầu.

2. Redis

Redis là kho lưu trữ dữ liệu NoSQL mã nguồn mở, được sử dụng phổ biến như bộ nhớ cache, hệ thống nhắn tin, cơ sở dữ liệu nhúng, v.v. Redis lưu trữ dữ liệu dưới dạng

cập key-value, với nhiều kiểu dữ liệu phong phú như string (chuỗi), hash (bảng băm), list (danh sách), set (tập hợp), sorted set (tập hợp sắp xếp).

Điểm nổi bật của Redis:

- **Tốc độ cao:** Redis lưu trữ dữ liệu trong bộ nhớ, cho phép truy cập và xử lý dữ liệu cực kỳ nhanh chóng, thường chỉ mất vài micro giây.
- **Dễ sử dụng:** Redis cung cấp giao diện lệnh đơn giản và giao thức truyền thông hiệu quả, giúp dễ dàng tích hợp vào ứng dụng.
- **Tính linh hoạt:** Redis hỗ trợ nhiều kiểu dữ liệu và cấu trúc dữ liệu, đáp ứng đa dạng nhu cầu lưu trữ.
- **Khả năng mở rộng:** Redis có thể dễ dàng mở rộng bằng cách thêm máy chủ mới, đáp ứng nhu cầu lưu trữ và truy cập dữ liệu ngày càng tăng.
- **Mã nguồn mở:** Redis miễn phí và mã nguồn mở, cho phép tùy chỉnh và phát triển theo nhu cầu.

1.3.3.4. Công cụ

1. Visual Studio Code

Visual Studio Code được biết đến là một trình biên tập lập trình code miễn phí dành cho Windows, Linux và macOS. Nó được phát triển bởi Microsoft là sự kết hợp hoàn hảo giữa IDE và Code Editor.

2. Nodejs

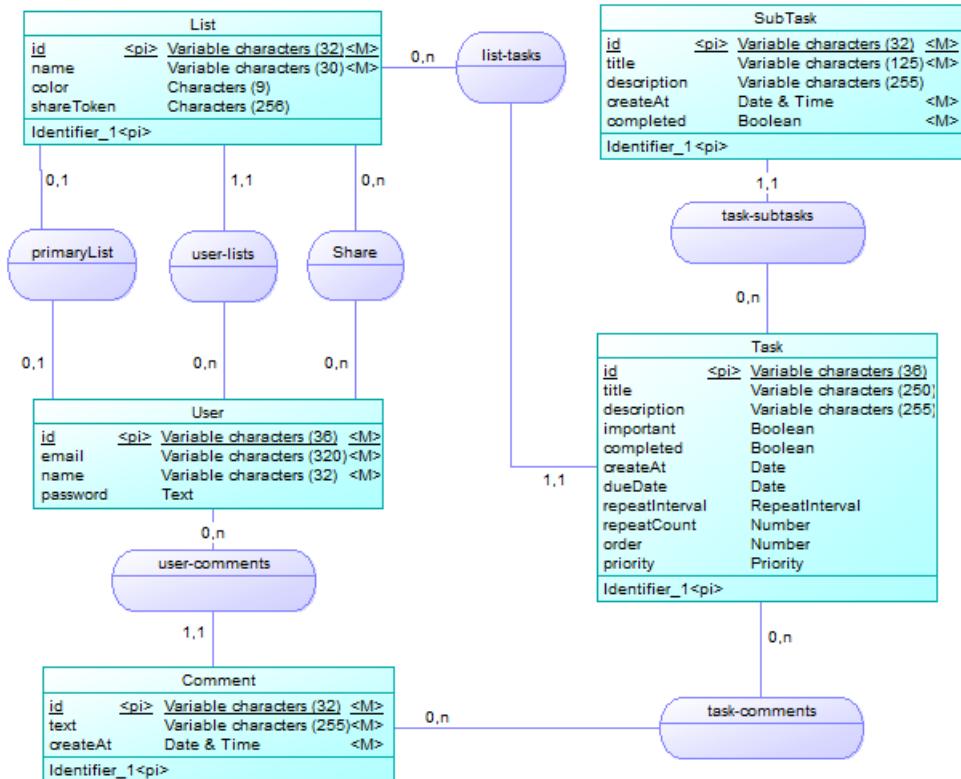
Node.js là một môi trường chạy mã JavaScript phía máy chủ được xây dựng trên nền tảng Chrome V8 JavaScript engine. Node.js có thể phát triển ứng dụng máy chủ hiệu suất cao, đáp ứng và có khả năng mở rộng một cách dễ dàng.

Một điểm đặc biệt của Node.js là nó sử dụng mô hình không đồng bộ (asynchronous) và sự kiện (event-driven) để xử lý các yêu cầu. Thay vì sử dụng luồng đồng bộ (synchronous) truyền thống, Node.js sử dụng mô hình không đồng bộ để xử lý nhiều yêu cầu cùng một lúc, giúp tối ưu hóa hiệu suất và khả năng đáp ứng của ứng dụng.

Node.js có thể xây dựng các ứng dụng web phức tạp, ứng dụng di động, ứng dụng IoT (Internet of Things), ứng dụng mạng xã hội và nhiều loại ứng dụng khác. Node.js cung cấp một cộng đồng mạnh mẽ và hỗ trợ đa dạng các thư viện và framework để phát triển ứng dụng dễ dàng.

CHƯƠNG 2: THIẾT KẾ CÁC MÔ HÌNH HỆ THỐNG

2.1. Mô hình quan niệm



Hình 1: Mô hình quan niệm

2.1. Mô hình luận lý

Mô hình tổ chức dữ liệu.

User(id, email, name, password, primaryListId)

List(id, name, color, shareToken, userId)

Share(listId, userId)

Task(id, title, important, completed, createdAt, repeatInterval, repeatCount, order, description, priority, listId)

SubTask(id, title, completed, description, createdAt, taskId)

Comment(id, text, createdAt, taskId, userId)

2.3. Mô hình Vật lý

User(id, email, name, password, primaryListId)

Tên thuộc tính	Kiểu dữ liệu	K thước	Ràng buộc toàn vẹn
Id (PK)	Varchar	Len()=36	
email	Varchar	Len()=320	
name	Varchar	Len()=32	
password	Text		
primaryListId(FK)	Varchar	Len()=36	Lookup(List)

Bảng 1: Mô hình vật lý User

List(id, name, color, shareToken, userId)

Tên thuộc tính	Kiểu dữ liệu	K thước	Ràng buộc toàn vẹn
Id (PK)	Varchar	Len()=36	
name	Varchar	Len()=32	
color	Char	Len()=9	
shareToken	Varchar	Len()=256	
userId (FK)	Varchar	Len()=36	Lookup(User)

Bảng 2: Mô hình vật lý List

Share(listId, userId)

Tên thuộc tính	Kiểu dữ liệu	K thước	Ràng buộc toàn vẹn
listId (PK, FK)	Varchar	Len()=36	Lookup(List)
userId (PK, FK)	Varchar	Len()=36	Lookup(User)

Bảng 3: Mô hình vật lý Share

Task(id, title, important, completed, createdAt, repeatInterval, repeatCount, order, description, priority, listId)

Tên thuộc tính	Kiểu dữ liệu	K thước	Ràng buộc toàn vẹn
id (PK)	Varchar	Len()=36	
title	Varchar	Len()=250	
description	Varchar	Len()=512	
important	Boolean		
completed	Boolean		
createdAt	Date		

repeatCount	Int		
repeatInterval	RepeatInterval		
order	Int		
Priority	Priority		
listId (FK)	Varchar	Len()=36	Lookup(List)

Bảng 4: Mô hình vật lý Task

SubTask(id, title, completed, description, createdAt, taskId)

Tên thuộc tính	Kiểu dữ liệu	K thước	Ràng buộc toàn vẹn
id (PK)	Varchar	Len()=36	
title	Varchar	Len()=125	
description	Varchar	Len()=255	
completed	Boolean		
taskId (FK)	Varchar	Len()=36	Lookup(Task)
createdAt	Date & time		

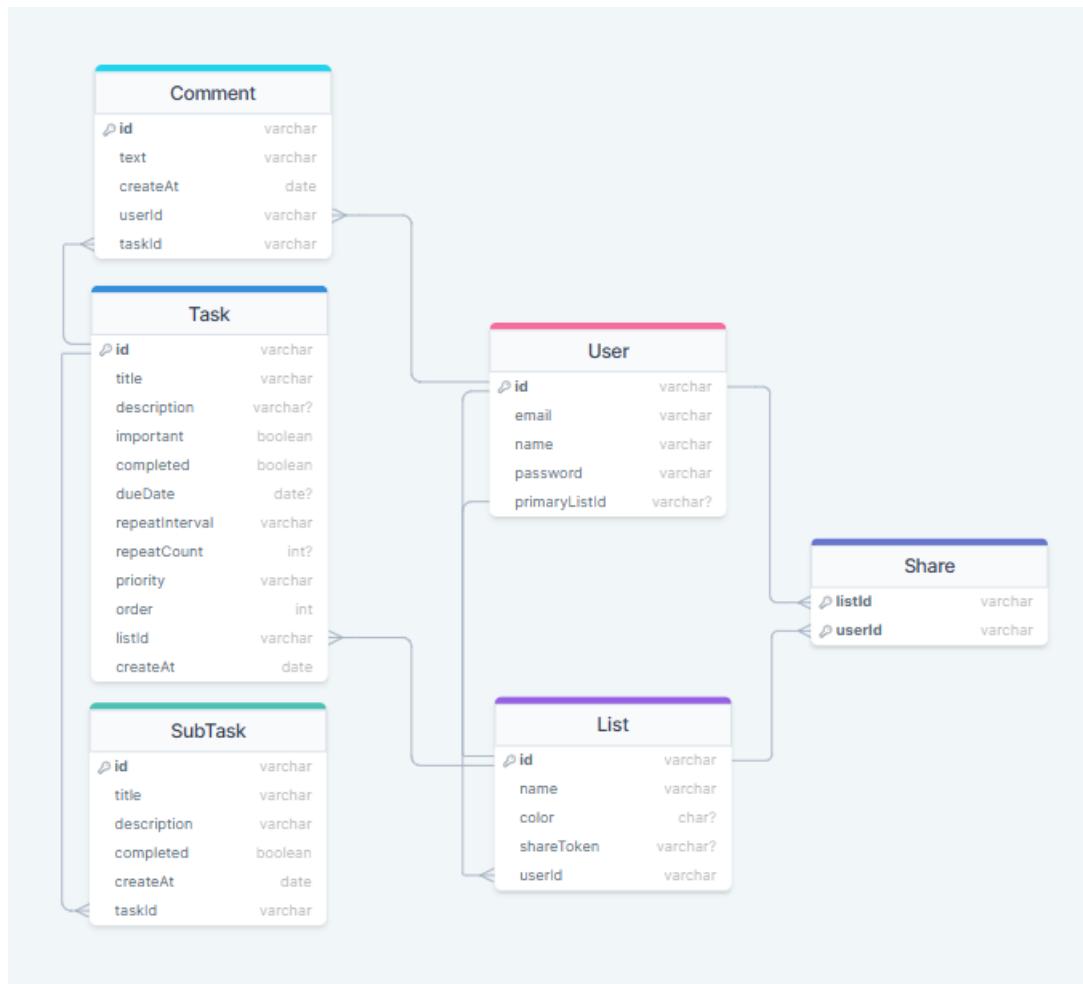
Bảng 5: Mô hình vật lý Subtask

Comment(id, text, createdAt, taskId, userId)

Tên thuộc tính	Kiểu dữ liệu	K thước	Ràng buộc toàn vẹn
id (PK)	Varchar	Len()=36	
text	Varchar	Len()=255	
createdAt	Date & time		
taskId (FK)	Varchar	Len()=36	Lookup(Task)
userId (FK)	Varchar	Len()=36	Lookup(User)

Bảng 6: Mô hình vật lý Comment

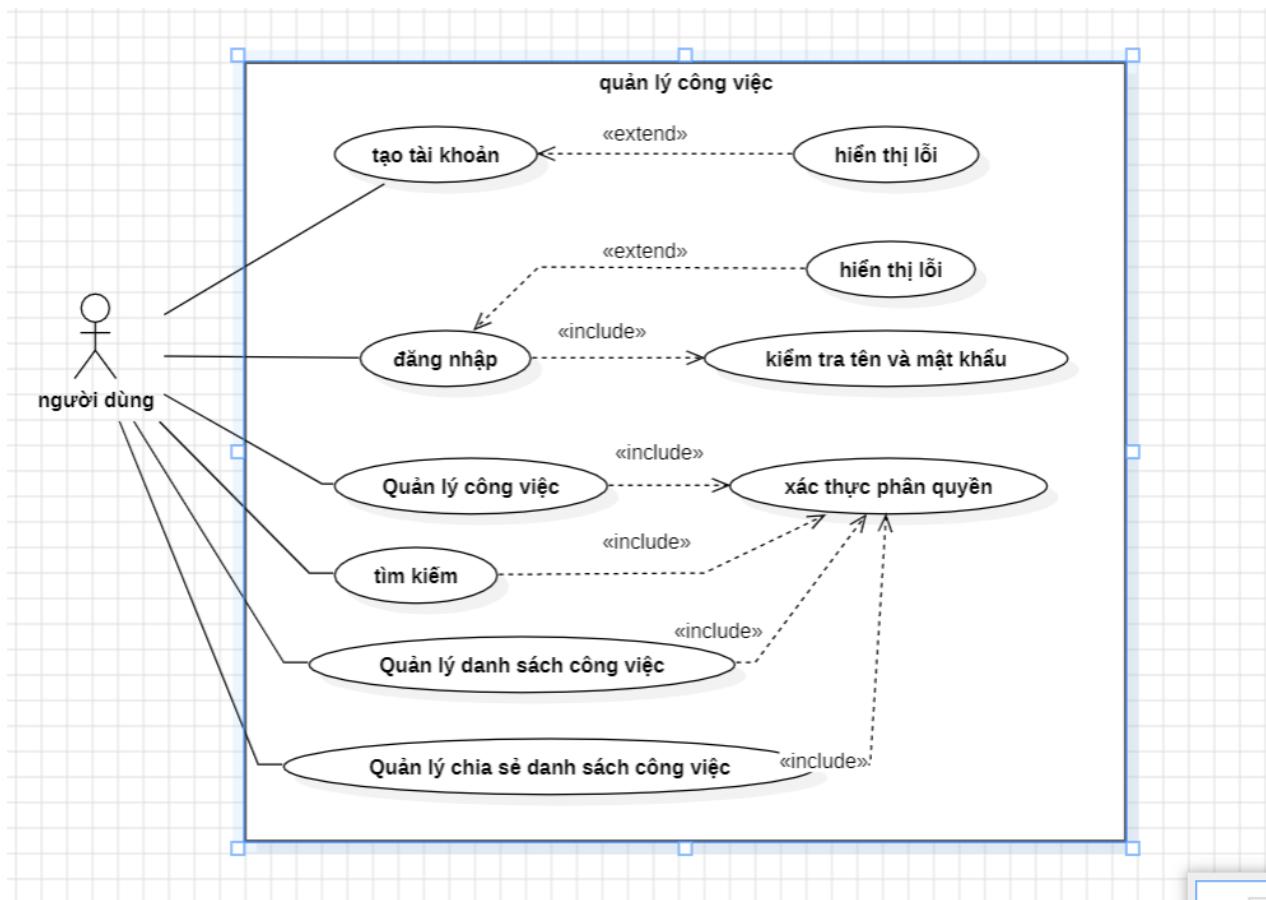
2.4. Mô hình liên kết các bảng trên máy tính



Hình 2: Mô hình liên kết các bảng trên máy tính

2.5. Mô hình xử lý (UML)

2.5.1. Mô hình use case

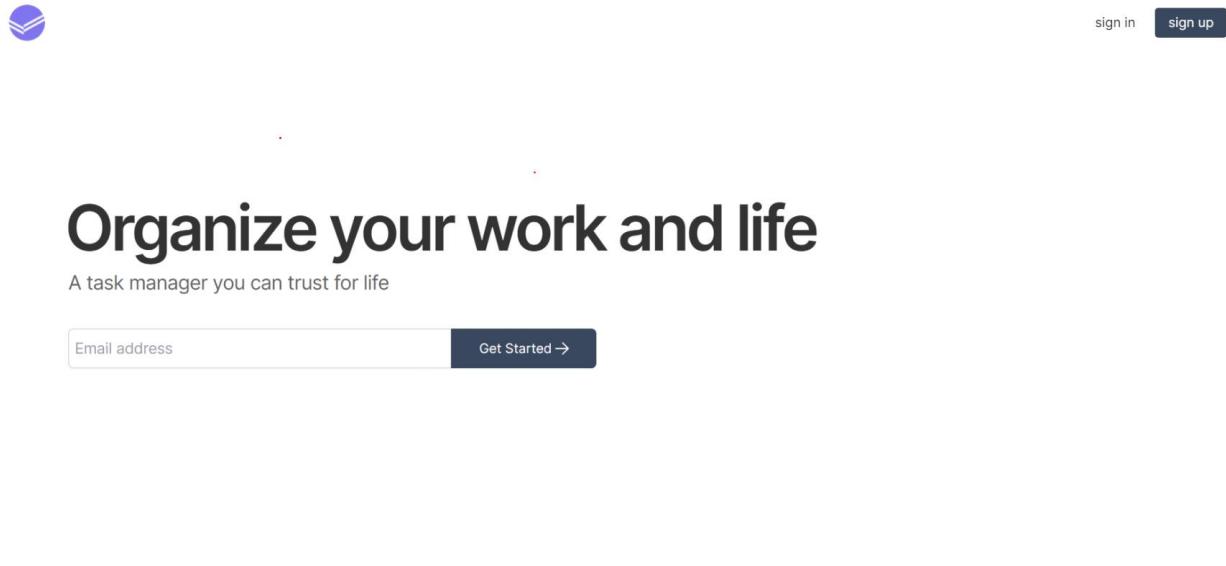


Hình 3: Mô hình usecase

CHƯƠNG 3: CÁC MỤC XỬ LÝ (DEMO)

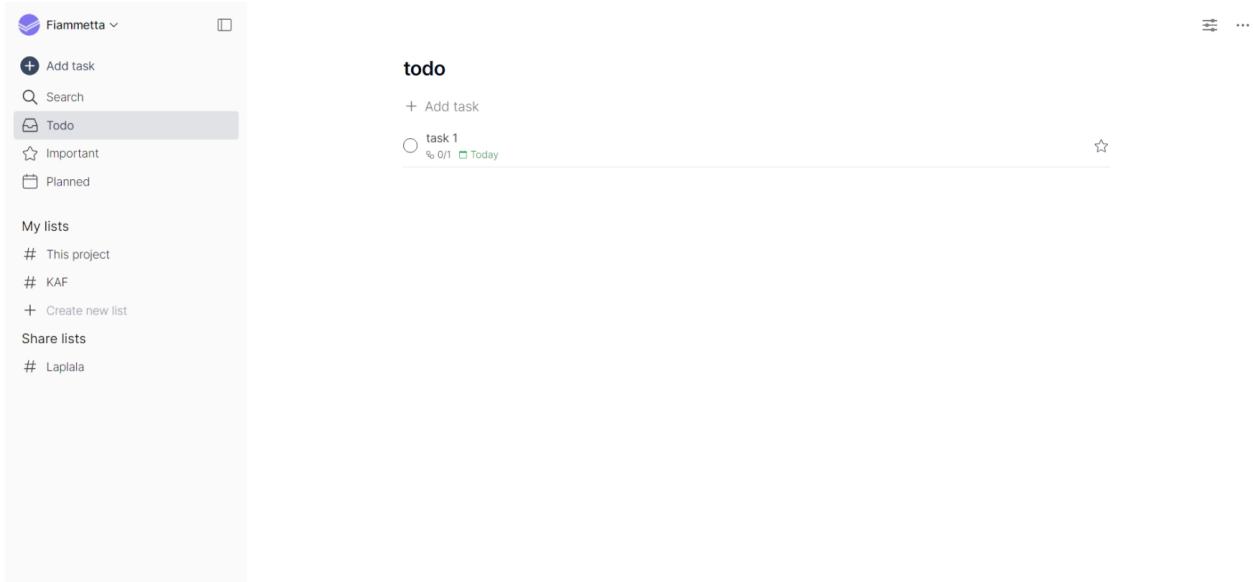
3.1. Trang chủ

Giao diện trang chủ:



Hình 4: Giao diện trang chủ

Giao diện ứng dụng:



Hình 5: Giao diện ứng dụng

3.2. Đăng nhập hệ thống

Giao diện đăng nhập:



Sign in

Username

Password

[Forgot password?](#)

Sign in

Do not have an account? [create new account](#)

Hình 6: Giao diện đăng nhập

Code chính đăng nhập:

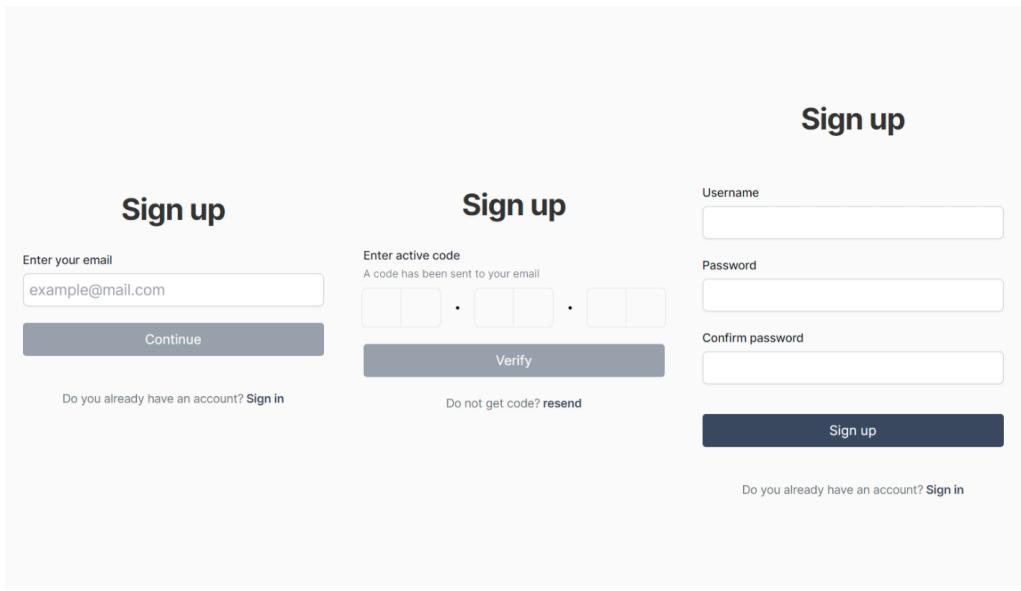
```
async authorize(credentials) {
    const parsedCredentials = z.object({
        username: z.string(),
        password: z.string(),
    }).safeParse(credentials);

    if (parsedCredentials.success) {
        const { username, password } = parsedCredentials.data;
        const user = await getUserByUsername(username);
        if (!user) return null;

        if (await bcrypt.compare(password, user.password)) {
            return {
                id: user.id,
                name: user.name,
                email: user.email,
            };
        }
    }
    return null;
}
```

3.2. Tạo tài khoản

Giao diện tạo tài khoản:



Hình 7: Giao diện tạo tài khoản

Code chính tạo tài khoản:

```

export async function createUser({
    username,
    email,
    password,
}: Account): Promise<[User, undefined] | [undefined, Error]> {
    const hashPassword = await bcrypt.hash(password, 10);

    try {
        const user = await prisma.user.create({
            data: {
                name: username,
                email: email,
                password: hashPassword,
                List: {
                    create: {
                        name: 'Home 🏠',
                    },
                },
            },
        });
        await getPrimaryList(user.id);

        return [user, undefined];
    } catch (error) {
        return [undefined, error as Error];
    }
}

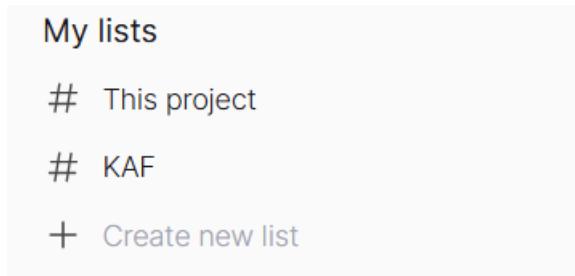
```

3.2. Cập nhật: thêm, sửa, xóa

3.2.1. Cập nhật list

3.2.1.1. Thêm

Giao diện thêm danh sách:



Hình 8: Giao diện thêm danh sách

Code chính:

```
export const createListHandler = factory.createHandlers(
  auth,
  zValidator('json', ListCreateSchema),
  async (c) => {
    const user = c.get('user');
    const body = c.req.valid('json');

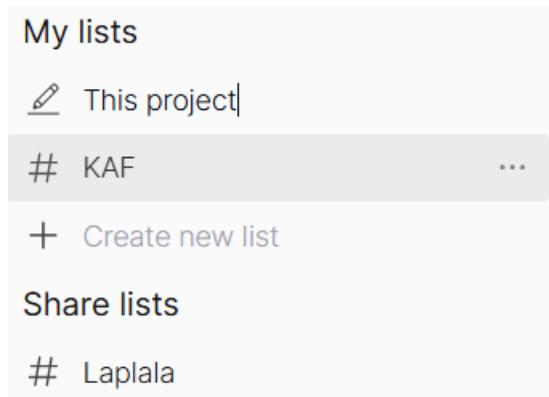
    const [data, err] = await withError(prisma.list.create)({
      data: {
        ...body,
        userId: user.id,
      },
    });

    if (!data) return c.json(undefined, 500);

    return c.json({ data: data });
  }
);
```

3.2.1.2. Sửa

Giao diện sửa danh sách:



Hình 9: Giao diện sửa danh sách

Code chính:

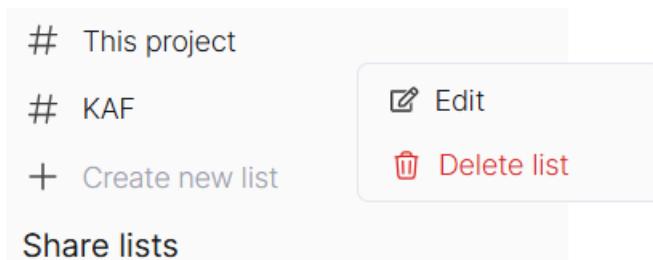
```
export const updateListHandler = factory.createHandlers(
  auth,
  zValidator('json', ListUpdateSchema),
  async (c) => {
    const body = c.req.valid('json');
    const user = c.get('user');
    const { id } = c.req.param();

    const [data, err] = await withError(prisma.list.update)({
      data: body,
      where: {
        id: id,
        userId: user.id,
      },
    });

    return c.json(undefined, err ? 401 : 204);
  }
);
```

3.2.1.3. Xóa

Giao diện xóa danh sách:



Hình 10: Giao diện xóa danh sách

Code chính:

```

export const deleteListHandler = factory.createHandlers(auth, async (c) => {
  const { id } = c.req.param();
  const user = c.get('user');

  const list = await prisma.list.findUnique({
    select: { id: true },
    where: { id, userId: user.id },
  });

  if (!list) return c.json(undefined, { status: 401 });

  const err = await deleteList(id);

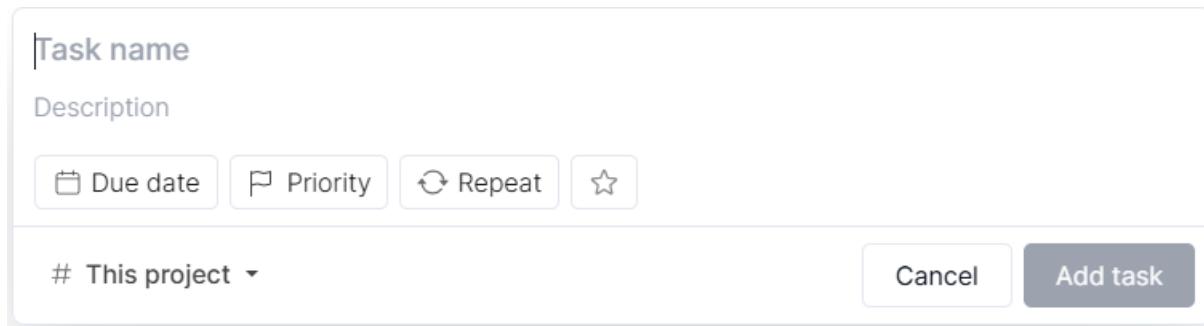
  return c.json(undefined, { status: err ? 401 : 204 });
});

```

3.2.2. Cập nhật công việc

3.2.2.1. Thêm

Giao diện thêm công việc:



Hình 11: Giao diện thêm công việc

Code chính:

```

export const createTaskHandler = factory.createHandlers(
  auth,
  zValidator('json', CreateTaskSchema),
  async (c) => {
    const user = c.get('user');
    const body = c.req.valid('json');

    const { subTasks, dueDate, ...task } = body;
    //check
    const [list] = await withError(prisma.list.findUnique)({
      where: {
        id: task.listId,
        OR: [
          {

```

```

        userId: user.id,
    },
    { Share: { some: { userId: user.id } } },
],
},
select: {
    id: true,
},
});
};

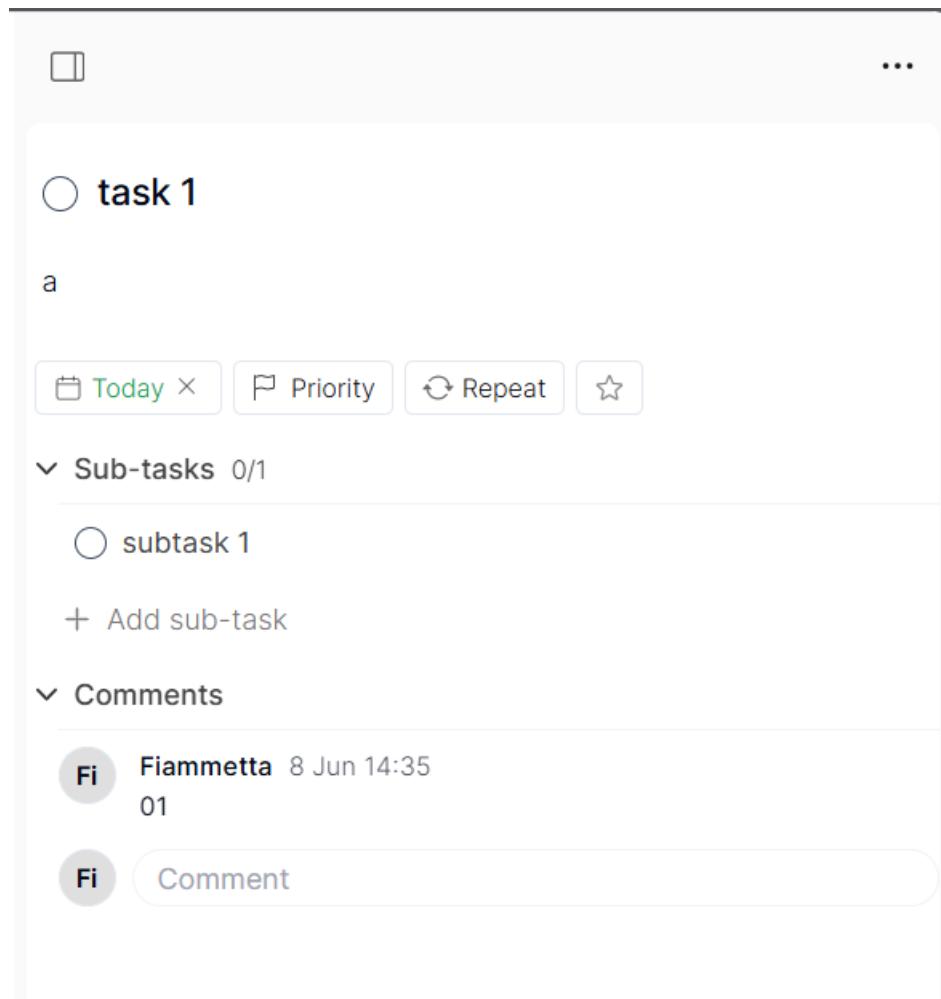
if (!list) return new Response(undefined, { status: 401 });
// create
const [data] = await withError(prisma.task.create)({
    data: {
        ...task,
        dueDate: convertTime(dueDate),
        subTasks: subTasks && {
            createMany: {
                data: subTasks,
            },
        },
        select: TASK_SELECT,
    });
}

if (data) return c.json({ data: data }, 200);
}
);

```

3.2.2.2. Sửa

Giao diện công việc:



Hình 12: Giao diện công việc

Code chính:

```
export const updateTaskHandler = factory.createHandlers(
  auth,
  zValidator('json', TaskUpdateSchema),
  async (c) => {
    const user = c.get('user');
    const id = c.req.param('id');
    const body = c.req.valid('json');

    const { subTasks, ...task } = body;

    try {
      await prisma.task.update({
        data: {
          ...task,
          dueDate: convertTime(body.dueDate),
        },
        where: {
          id: id,
        }
      })
    } catch (err) {
      return c.status(500).json({ error: err.message })
    }
  }
)
```

```

        list: {
          OR: [
            {
              userId: user.id,
            },
            { Share: { some: { userId: user.id } } },
          ],
        },
      },
    });
  });

  return c.json(undefined, 204);
} catch (error) {
  return c.json(undefined, 410);
}
}

);

```

3.2.2.3. Xóa

Code chính:

```

export const deleteTaskHandler = factory.createHandlers(auth, async (c) => {
  const user = c.get('user');
  const id = c.req.param('id');

  const [task] = await findTaskById(
    { taskId: id, userId: user.id },
    { id: true }
  );

  if (!task) return c.json(undefined, 401);

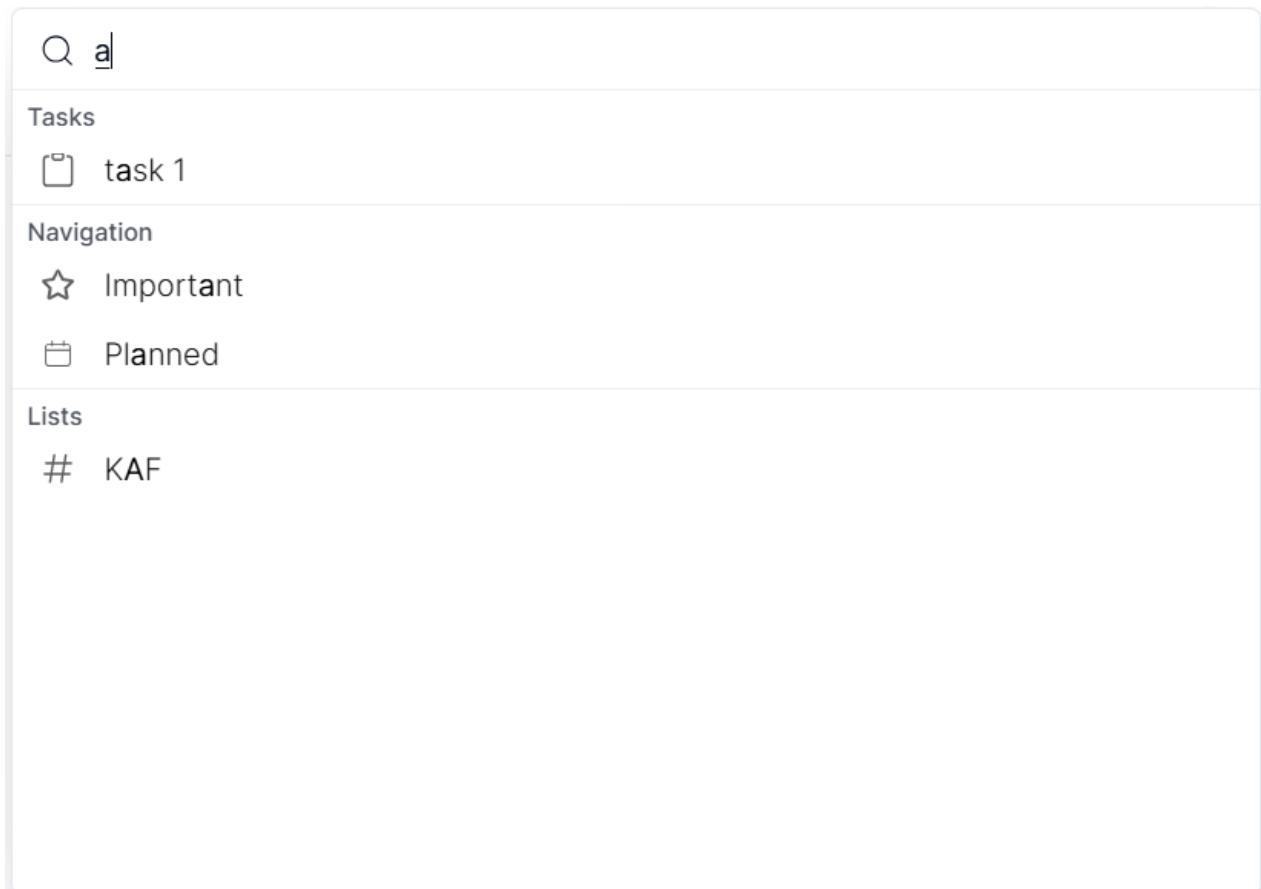
  const err = await deleteTaskById(id);

  return c.json(undefined, err ? 401 : 204);
});

```

3.3. Tìm kiếm, hiển thị, xử lý thông tin theo yêu cầu đề tài

Giao diện tìm kiếm:



Hình 13: Giao diện tìm kiếm

Giao diện kết quả:

Results for "a"

- Mostima 99% 0/1 ★
- Data ★
- task 1 99% 0/1 Today ★
- completed 1
- an.com 99% 0/1 5 Jun ★

A list of search results for the query "a". The results are displayed in a card-based format. Each card shows the name of the result, its completion percentage (99%), the number of tasks (0/1), the date (Today or 5 Jun), and a star icon for rating. The first three results have a red circular icon with a checkmark to their left, while the last two have a green circular icon with a checkmark.

Hình 14: Giao diện kết quả tìm kiếm

Code chính:

```
export const searchTasksHandler = factory.createHandlers(auth, async (c) => {
  const user = c.get('user');
  const q = c.req.param('q');
```

```

const tasks = await prisma.task.findMany({
    select: TASK_SELECT,
    where: {
        title: {
            contains: q,
            mode: 'insensitive',
        },
        list: {
            OR: [
                {
                    userId: user.id,
                },
                { Share: { some: { userId: user.id } } },
            ],
        },
    },
});
if (tasks.length > 0)
    return c.json({
        data: {
            tasks,
        },
    });
return c.json(undefined, 404);
});

```

3.4. Thống kê, tổng hợp số liệu theo yêu cầu đề tài

Giao diện xem danh sách:

The screenshot shows a task management application interface. On the left, there's a sidebar with a profile icon and the name 'Fiammetta'. It includes sections for 'Add task', 'Search', 'Todo', 'Important', 'Planned', and 'My lists'. Under 'My lists', '# This project' is highlighted. Other lists include '# KAF', '+ Create new list', 'Share lists', and '# Laplala'. The main content area is titled 'This project' and shows a list of tasks: 'ăn cơm', 'Data', and 'Mostima'. Each task has a progress bar indicating completion status. A right sidebar contains fields for 'Description', 'Due date', 'Priority', 'Repeat', and a 'Comment' input field.

Hình 15: Giao diện xem danh sách

Code chính:

```
export const getTaskHandler = factory.createHandlers(auth, async (c) => {
  const user = c.get('user');
  const id = c.req.param('id');

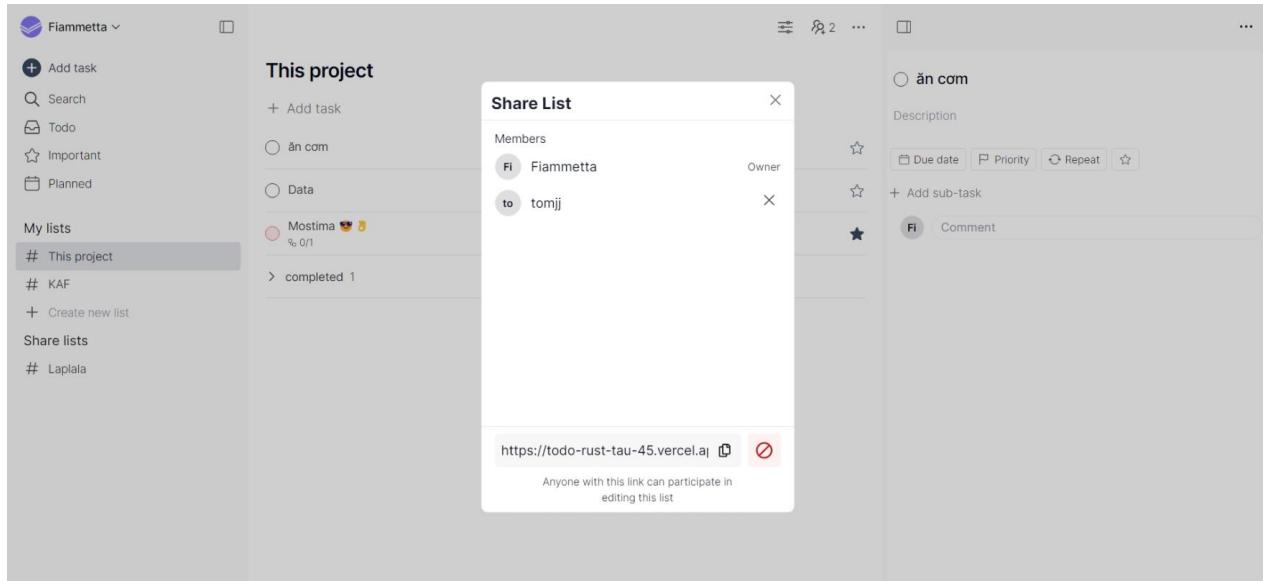
  const tasks = await noError(prisma.task.findUnique)({
    where: {
      id: id,
      list: {
        OR: [
          { userId: user.id },
          { Share: { some: { userId: user.id } } },
        ],
      },
    },
    select: TASK_SELECT,
  });

  if (!tasks) return c.json(undefined, 410);

  return c.json({ data: tasks });
});
```

3.5. Chia sẻ danh sách

Giao diện chia sẻ danh sách:



Hình 16: Giao diện chia sẻ danh sách

Code chính:

```
export const createNewShareListToken = factory.createHandlers(
```

```

auth,
async (c) => {
  const user = c.get('user');
  const listId = c.req.param('id');

  const [primary, err] = await getPrimaryList(user.id, { id: true });
  if (err) return c.json(undefined, 500);

  const token = createRandString(80);

  try {
    const list = await prisma.list.update({
      select: {
        shareToken: true,
      },
      data: {
        shareToken: token,
      },
      where: {
        id: listId,
        userId: user.id,
        NOT: {
          id: primary.id,
        },
      },
    });
    return c.json({ data: list });
  } catch (err) {
    return c.json(undefined, 500);
  }
};

async function Page({ params, searchParams }: Props) {
  const user = await getSessionUser();

  if (!searchParams.InvitationTokens || !user) return ErrorPage(400);

  try {
    await new Promise((resolve) => setTimeout(resolve, 200));

    const data = await prisma.list.findUnique({
      select: {
        id: true,
      },
      where: {
        id: params.id,
      },
    });
  
```

```
shareToken: searchParams.InvitationTokens,  
  
        NOT: {  
            userId: user.id,  
        },  
    },  
});  
  
if (!data) throw new Error('not found');  
  
await prisma.share.create({  
    data: {  
        userId: user.id,  
        listId: params.id,  
    },  
});  
} catch (error) {  
    return ErrorPage(500);  
}  
  
redirect(`/tasks/${params.id}`);  
return null;  
}
```

CHƯƠNG 4: KẾT LUẬN VÀ KIẾN NGHỊ

4.1. Kết quả đạt được

Trong quá trình tìm hiểu và khảo sát trên, em đã đạt được một số kết quả sau:

- Phân tích được quy trình hoạt động và các chức năng của hệ thống.
- Xây dựng được các chức năng của ứng dụng web quản lý công việc.

Phần mềm đã xây dựng được các tính năng chính như:

- Chức năng tạo tài khoản.
- Chức năng đăng nhập.
- Quản lý các công việc.
- Quản lý các danh sách công việc.
- Chia sẻ các danh sách công việc cho người dùng khác.

Ưu điểm của ứng dụng:

- Dễ sử dụng.
- Giao diện thân thiện.

Nhược điểm của phần mềm:

- Thiết kế hệ thống chưa được tốt.
- Ít khả năng mở rộng.
- Chưa hỗ trợ thời gian thực.

4.2. Hướng phát triển đề tài

Từ những kết quả đề tài trên, cần có một hướng phát triển mới để đề tài ngày càng hoàn thiện hơn:

- Tiếp tục nghiên cứu, xây dựng ứng dụng tốt hơn.
- Thêm các thiệu ứng âm thanh.
- Tối ưu hóa hệ thống.
- Sửa các lỗi tồn động.
- Thêm tính năng thời gian thực cho các danh sách được chia sẻ.
- Thêm các sáp xếp theo và lọc theo.
- Thêm tính năng cá nhân hóa.

- Phát triển ứng dụng cho mobile.

4.3. Kiến nghị

Do vốn hiểu biết của chúng em còn nhiều hạn chế nên chương trình còn rất nhiều hạn chế. Chính vì vậy, chúng em mong thầy hướng dẫn cho đè tài của chúng em thêm.

TÀI LIỆU THAM KHẢO

<https://developer.mozilla.org/en-US/docs/Web>

<https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/>

<https://aws.amazon.com/vi/what-is/javascript/>

<https://www.typescriptlang.org/>

<https://redis.io/about/>

<https://www.mordorintelligence.com/industry-reports/task-management-software-market>

<https://nodejs.org/en/about>

Kiên Giang, ngày tháng năm

Sinh viên thực hiện