

## **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*Vĩnh Long, ngày 05 tháng 01 năm 2026*  
**Giảng viên hướng dẫn**  
(Ký tên và ghi rõ họ tên)

# Nguyễn Hoàng Duy Thiện

# **NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Vĩnh Long, ngày 05 tháng 01 năm 2026*  
**Thành viên hội đồng**  
(*Ký tên và ghi rõ họ tên*)

## LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời tri ân sâu sắc đến quý thầy cô, giảng viên Trường Đại học Trà Vinh, đặc biệt là các thầy cô thuộc Khoa Công nghệ Thông tin, Trường Kỹ thuật & Công nghệ đã tạo điều kiện thuận lợi để em có thể hoàn thành bài báo cáo này một cách trọn vẹn.

Và em cũng xin bày tỏ lòng biết ơn chân thành đến thầy Nguyễn Hoàng Duy Thiện – Giảng viên Khoa Công nghệ Thông tin, Trường Đại học Trà Vinh, người đã tận tình hướng dẫn em trong suốt quá trình thực hiện đồ án. Thầy không chỉ chia sẻ những kiến thức chuyên môn quý báu mà còn truyền đạt cho em tinh thần học hỏi và sự đam mê trong lĩnh vực Công nghệ Thông tin.

Vì những kinh nghiệm và kiến thức của em còn hạn chế, bài báo cáo chắc chắn khó tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và góp ý quý báu từ thầy cô để có thể hoàn thiện hơn trong những nghiên cứu sau này.

Cuối cùng, em xin kính chúc quý thầy cô luôn dồi dào sức khỏe và tràn đầy nhiệt huyết trong sự nghiệp giảng dạy.

Em xin chân thành cảm ơn!

**Sinh viên thực hiện**

**Trầm Khôi Nguyên**

## MỤC LỤC

CHƯƠNG 1: TỔNG QUAN .....	13
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT .....	14
2.1 Next.js .....	14
2.1.1 Giới thiệu .....	14
2.1.2 Ưu điểm của Next.js .....	14
2.1.3 Nhược điểm của Next.js .....	15
2.1.4 Một số đặc điểm cơ bản trong Next.js .....	15
2.2 ExpressJS .....	17
2.2.1 Giới thiệu .....	17
2.2.2 Ưu điểm của ExpressJS .....	17
2.2.3 Nhược điểm của ExpressJS .....	17
2.2.4 Một số tính năng của ExpressJS .....	18
2.3 Node.js .....	18
2.3.1 Giới thiệu .....	18
2.3.2 Ưu điểm của Node.js .....	19
2.3.3 Nhược điểm của Node.js .....	19
2.3.4 Một số đặc điểm cơ bản của Node.js .....	20
2.4 Clean Architecture .....	20
2.4.1 Giới thiệu .....	20
2.4.2 Ưu điểm của Clean Architecture .....	21
2.4.3 Nhược điểm của Clean Architecture .....	21
2.4.4 Một số đặc điểm cơ bản trong Clean Architecture .....	22
2.5 RESTful API .....	24
2.5.1 Giới thiệu .....	24
2.5.2 Nguyên tắc cơ bản của RESTful API .....	25
2.5.3 Ưu điểm của RESTful API .....	26
2.5.4 Nhược điểm RESTful API .....	27
2.6 MySQL .....	27
2.6.1 Giới thiệu .....	27
2.6.2 Ưu điểm của MySQL .....	28
2.6.3 Nhược điểm MySQL .....	28
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU .....	29
3.1 Mô tả bài toán .....	29

3.2 Đặc tả yêu cầu hệ thống.....	29
3.2.1 Yêu cầu chức năng .....	29
3.2.2 Yêu cầu phi chức năng .....	31
3.3 Sơ đồ phân cấp chức năng .....	31
3.4 Sơ đồ Use-case .....	32
3.5 Kiến trúc tổng thể của hệ thống.....	33
3.6 Thiết kế thành phần dữ liệu .....	34
3.6.1 Mô hình dữ liệu mức quan niệm .....	34
3.6.2 Mô hình dữ liệu mức luận lý .....	35
3.6.3 Mô hình dữ liệu mức vật lý .....	35
3.6.4 Lược đồ cơ sở dữ liệu.....	36
3.6.5 Thiết kế cơ sở dữ liệu .....	36
3.7 Thiết kế giao diện .....	42
3.7.1 Giao diện trang chủ .....	42
3.7.2 Giao diện trang tìm việc nhanh .....	42
3.7.3 Giao diện trang tìm việc .....	43
3.7.4 Giao diện trang tìm kiếm công ty .....	44
3.7.5 Giao diện trang chi tiết công việc.....	44
3.7.6 Giao diện trang chi tiết đơn ứng tuyển .....	45
3.7.7 Giao diện trang quản lý đơn ứng tuyển .....	46
3.7.8 Giao diện trang đăng bài tuyển dụng.....	46
3.7.9 Giao diện trang đánh giá CV với AI .....	47
3.8 Triển khai.....	48
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU .....	51
4.1 Kết quả thử nghiệm API.....	51
4.1.1 Các API xác thực .....	51
4.1.2 Các API ứng tuyển việc làm.....	53
4.1.3 Các API công việc .....	55
4.1.4 Các API ngành nghề .....	57
4.1.5 Các API nhà tuyển dụng.....	58
4.1.6 Các API người tìm việc .....	60
4.1.7 API đánh giá CV với AI .....	61
4.2 Các giao diện của hệ thống.....	62
4.2.1 Giao diện đăng nhập người tìm việc .....	62

4.2.2 Giao diện đăng ký người tìm việc .....	63
4.2.3 Giao diện đăng nhập nhà tuyển dụng .....	63
4.2.4 Giao diện đăng ký nhà tuyển dụng .....	64
4.2.5 Giao diện trang chủ .....	64
4.2.6 Giao diện trang ngành nghề, địa điểm.....	65
4.2.7 Giao diện trang tìm kiếm việc làm .....	66
4.2.8 Giao diện trang tìm kiếm công ty .....	66
4.2.9 Giao diện trang chi tiết công việc.....	67
4.2.10 Giao diện trang chi tiết nhà tuyển dụng .....	68
4.2.11 Giao diện trang chi tiết người tìm việc.....	69
4.2.12 Giao diện trang quản lý đơn ứng tuyển .....	69
4.2.13 Giao diện trang chi tiết đơn ứng tuyển .....	70
4.2.14 Giao diện trang đánh giá CV với AI .....	71
<b>CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>72</b>
5.1 Kết luận.....	72
5.2 Hướng phát triển.....	72
<b>DANH MỤC TÀI LIỆU THAM KHẢO .....</b>	<b>73</b>

## **DANH MỤC HÌNH ẢNH – BẢNG BIỂU**

### **Danh mục hình ảnh**

Hình 2.1 Clean Architecture.....	22
Hình 3.1 Sơ đồ phân cấp chức năng .....	31
Hình 3.2: Sơ đồ Use-case .....	32
Hình 3.3: Kiến trúc hệ thống.....	33
Hình 3.4: Mô hình dữ liệu mức quan niệm .....	34
Hình 3.5: Mô hình dữ liệu mức luận lý .....	35
Hình 3.6: Mô hình dữ liệu mức vật lý .....	35
Hình 3.7: Lược đồ cơ sở dữ liệu.....	36
Hình 3.8: Thiết kế giao diện trang chủ.....	42
Hình 3.9: Thiết kế giao diện trang tìm việc nhanh.....	43
Hình 3.10: Thiết kế giao diện trang tìm việc.....	43
Hình 3.11: Thiết kế giao diện trang tìm kiếm công ty.....	44
Hình 3.12: Thiết kế giao diện trang chi tiết công việc .....	45
Hình 3.13: Thiết kế giao diện trang chi tiết đơn ứng tuyển .....	45
Hình 3.14: Thiết kế giao diện trang quản lý đơn ứng tuyển .....	46
Hình 3.15: Thiết kế giao diện trang đăng bài tuyển dụng .....	47
Hình 3.16: Thiết kế giao diện trang đánh giá CV với AI .....	47
Hình 4.1: Thủ nghiệm API tạo tài khoản người tìm việc .....	51
Hình 4.2: Thủ nghiệm API đăng nhập tài khoản người tìm việc .....	52
Hình 4.3 Thủ nghiệm API tạo tài khoản nhà tuyển dụng.....	52
Hình 4.4 Thủ nghiệm API đăng nhập tài khoản nhà tuyển dụng.....	53
Hình 4.5: Thủ nghiệm API ứng tuyển công việc .....	54
Hình 4.6: Thủ nghiệm API lấy danh sách đơn ứng tuyển của người tìm việc .....	54
Hình 4.7: Thủ nghiệm API hủy ứng tuyển công việc .....	55
Hình 4.8: Thủ nghiệm API tìm kiếm công việc .....	56
Hình 4.9: Thủ nghiệm API tạo công việc cho nhà tuyển dụng .....	56
Hình 4.10: Thủ nghiệm API xóa công việc cho nhà tuyển dụng.....	57
Hình 4.11: Thủ nghiệm API lấy danh sách ngành nghề .....	58
Hình 4.12: Thủ nghiệm API lấy danh sách ngành nghề theo loại .....	58
Hình 4.13: Thủ nghiệm API lấy danh sách tất cả công ty .....	59
Hình 4.14: Thủ nghiệm API lấy thông tin công ty hiện tại .....	59
Hình 4.15: Thủ nghiệm API cập nhật thông tin công ty .....	60

Hình 4.16: Thủ nghiệm API lấy thông tin người tìm việc theo ID .....	60
Hình 4.17: Thủ nghiệm API cập nhật thông tin người tìm việc .....	61
Hình 4.18: Thủ nghiệm API lấy thông tin người tìm việc hiện tại .....	61
Hình 4.19: Thủ nghiệm API đánh giá hồ sơ xin việc với AI .....	62
Hình 4.20: Giao diện trang đăng nhập tài khoản người tìm việc .....	63
Hình 4.21: Giao diện trang đăng ký tài khoản người tìm việc .....	63
Hình 4.22: Giao diện trang đăng nhập tài khoản nhà tuyển dụng .....	64
Hình 4.23: Giao diện trang đăng ký tài khoản nhà tuyển dụng .....	64
Hình 4.24: Giao diện trang chủ .....	65
Hình 4.25: Giao diện trang ngành nghề, địa điểm .....	66
Hình 4.26: Giao diện trang tìm kiếm việc làm .....	66
Hình 4.27: Giao diện trang tìm kiếm công ty .....	67
Hình 4.28: Giao diện trang chi tiết công việc .....	68
Hình 4.29: Giao diện trang chi tiết nhà tuyển dụng .....	68
Hình 4.30: Giao diện trang chi tiết người tìm việc .....	69
Hình 4.31: Giao diện trang quản lý đơn ứng tuyển .....	70
Hình 4.32: Giao diện trang chi tiết đơn ứng tuyển .....	70
Hình 4.33: Giao diện trang đánh giá CV với AI .....	71

### Danh mục bảng biểu

Bảng 3.1: Chi tiết thực thể “users” .....	36
Bảng 3.2: Chi tiết thực thể “provinces” .....	37
Bảng 3.3: Chi tiết thực thể “fields” .....	37
Bảng 3.4: Chi tiết thực thể “companies” .....	38
Bảng 3.5: Chi tiết thực thể “jobs” .....	38
Bảng 3.6: Chi tiết thực thể “apply_job” .....	39
Bảng 3.7: Chi tiết thực thể “save_job” .....	40
Bảng 3.8: Chi tiết thực thể “follow_company” .....	41
Bảng 3.9: Chi tiết thực thể “notifications” .....	41

## TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

### Vấn đề nghiên cứu

Trong bối cảnh chuyển đổi số ngày càng phát triển, nhu cầu tìm kiếm và ứng tuyển việc làm trực tuyến của người lao động và nhu cầu tuyển dụng của doanh nghiệp ngày càng gia tăng. Tuy nhiên, nhiều nền tảng hiện nay vẫn còn hạn chế về khả năng mở rộng, trải nghiệm người dùng và tính linh hoạt trong quản lý hệ thống.

Xuất phát từ thực tế đó, đồ án này tập trung nghiên cứu và xây dựng một website tìm kiếm và ứng tuyển việc làm trực tuyến, nhằm hỗ trợ kết nối hiệu quả giữa người tìm việc và nhà tuyển dụng. Để tài ứng dụng các công nghệ hiện đại như Next.js cho frontend, Node.js kết hợp Express.js cho backend và MySQL cho cơ sở dữ liệu. Đồng thời, mô hình Clean Architecture được lựa chọn làm định hướng thiết kế chính nhằm đảm bảo tính rõ ràng trong cấu trúc hệ thống, nâng cao khả năng bảo trì và mở rộng trong tương lai.

### Các hướng tiếp cận

Đồ án được triển khai theo hướng tiếp cận dựa trên việc kết hợp các công nghệ web hiện đại và kiến trúc phần mềm chuẩn hóa. Trong giai đoạn nghiên cứu lý thuyết, tập trung tìm hiểu cơ chế hoạt động của Next.js, bao gồm server-side rendering và client-side rendering nhằm tối ưu hiệu năng và trải nghiệm người dùng.

Song song đó, việc xây dựng hệ thống backend bằng Node.js và Express.js theo kiến trúc RESTful API được nghiên cứu kỹ lưỡng để đảm bảo khả năng giao tiếp hiệu quả giữa frontend và backend. Trên cơ sở đó, mô hình Clean Architecture được áp dụng nhằm phân tách rõ ràng các lớp nghiệp vụ, giảm sự phụ thuộc giữa các thành phần trong hệ thống.

### Cách giải quyết vấn đề

Để giải quyết bài toán đặt ra, đồ án được thực hiện thông qua việc tổng hợp kiến thức từ tài liệu chính thức của Next.js, Node.js, Express.js và MySQL, kết hợp với việc tham khảo các nền tảng tuyển dụng trực tuyến phổ biến hiện nay nhằm phân tích yêu cầu thực tế của hệ thống.

Quá trình xây dựng hệ thống được tiến hành theo từng giai đoạn, bao gồm phân tích yêu cầu, thiết kế cơ sở dữ liệu, xây dựng giao diện người dùng và phát triển các

API backend. Trong đó, Next.js được sử dụng để phát triển giao diện thân thiện, responsive cho cả ứng viên và nhà tuyển dụng; Express.js đảm nhiệm xử lý nghiệp vụ và cung cấp các API; MySQL được sử dụng để quản lý dữ liệu liên quan đến người dùng, nhà tuyển dụng, công việc và đơn ứng tuyển. Việc áp dụng Clean Architecture giúp hệ thống dễ kiểm thử, dễ bảo trì và hạn chế lỗi phát sinh trong quá trình mở rộng.

### **Một số kết quả đạt được**

Kết quả của đồ án là một website tìm kiếm và ứng tuyển việc làm hoàn chỉnh, đáp ứng đầy đủ các chức năng cơ bản cho cả người tìm việc và nhà tuyển dụng. Giao diện được xây dựng bằng Next.js kết hợp Tailwind CSS và Shadcn UI, mang lại trải nghiệm hiện đại, trực quan và tương thích trên nhiều thiết bị.

Phần backend được phát triển theo kiến trúc Clean Architecture với các RESTful API rõ ràng, hỗ trợ tốt cho việc mở rộng và tích hợp trong tương lai. Hệ thống cơ sở dữ liệu MySQL được thiết kế chặt chẽ, đảm bảo tính toàn vẹn và hiệu quả trong truy xuất dữ liệu.

## MỞ ĐẦU

### Lý do chọn đề tài

Trong bối cảnh thị trường lao động ngày càng cạnh tranh và chuyển đổi số diễn ra mạnh mẽ, nhu cầu tìm kiếm việc làm và tuyển dụng trực tuyến ngày càng gia tăng. Các nền tảng tuyển dụng trực tuyến đã và đang trở thành cầu nối quan trọng giữa người tìm việc và nhà tuyển dụng, giúp tiết kiệm thời gian, chi phí và nâng cao hiệu quả tuyển dụng. Tuy nhiên, nhiều hệ thống hiện nay vẫn còn tồn tại những hạn chế về khả năng mở rộng, hiệu năng, trải nghiệm người dùng và tính linh hoạt trong quản lý dữ liệu.

Xuất phát từ thực tế đó, đề tài xây dựng website tìm kiếm và ứng tuyển việc làm được lựa chọn nhằm nghiên cứu và phát triển một hệ thống hỗ trợ hiệu quả cho quá trình tìm kiếm, ứng tuyển và quản lý việc làm trực tuyến. Việc ứng dụng các công nghệ hiện đại như Next.js, Node.js với Express.js cùng với cơ sở dữ liệu MySQL, kết hợp mô hình Clean Architecture, được xem là hướng tiếp cận phù hợp để xây dựng một hệ thống có tính ổn định, dễ bảo trì và khả năng mở rộng cao.

### Mục đích

Mục tiêu chính của đồ án là xây dựng một website tìm kiếm và ứng tuyển việc làm trực tuyến hoạt động ổn định, an toàn và thân thiện với người dùng, đáp ứng nhu cầu của cả ứng viên và nhà tuyển dụng.

Bên cạnh đó, đồ án còn nhằm giúp nắm vững và áp dụng hiệu quả các công nghệ phát triển web hiện đại như Next.js cho frontend, Express.js cho backend và MySQL cho cơ sở dữ liệu. Thông qua việc áp dụng mô hình Clean Architecture, đồ án hướng tới việc xây dựng một hệ thống tối ưu về cấu trúc, hiệu năng và bảo mật, đồng thời tạo nền tảng cho việc phát triển và mở rộng trong tương lai.

### Đối tượng

Đối tượng nghiên cứu của đề tài là hệ thống website tìm kiếm và ứng tuyển việc làm, cùng với các công nghệ và kiến trúc phần mềm hiện đại được sử dụng trong quá trình xây dựng hệ thống. Cụ thể, đề tài tập trung nghiên cứu cách ứng dụng Next.js trong phát triển giao diện người dùng, Node.js và Express.js trong xây dựng RESTful

API, MySQL trong quản lý dữ liệu, và Clean Architecture trong thiết kế kiến trúc hệ thống nhằm đảm bảo tính rõ ràng, linh hoạt và dễ bảo trì.

### **Phạm vi nghiên cứu**

Phạm vi nghiên cứu của đồ án tập trung vào việc xây dựng một website tìm kiếm và ứng tuyển việc làm với các chức năng cốt lõi như: quản lý tài khoản ứng viên và nhà tuyển dụng, tìm kiếm và lọc việc làm, đăng tin tuyển dụng, ứng tuyển trực tuyến, theo dõi trạng thái hồ sơ và quản lý thông tin việc làm.

Đồ án không đi sâu vào các chức năng nâng cao như thanh toán trực tuyến hay tích hợp hệ thống đánh giá chuyên sâu, mà chủ yếu tập trung vào việc thiết kế và triển khai một hệ thống hoàn chỉnh, sử dụng Next.js để xây dựng giao diện người dùng thân thiện và responsive, Express.js theo mô hình Clean Architecture để phát triển backend, và MySQL để quản lý cơ sở dữ liệu một cách hiệu quả.

## CHƯƠNG 1: TỔNG QUAN

Trong bối cảnh phát triển mạnh mẽ của công nghệ thông tin và Internet, các hoạt động tìm kiếm việc làm và tuyển dụng đang dần chuyển dịch sang môi trường trực tuyến. Sự thay đổi này xuất phát từ nhu cầu tối ưu hóa thời gian, chi phí và nâng cao hiệu quả kết nối giữa người lao động và doanh nghiệp. Các nền tảng tuyển dụng trực tuyến ra đời đã góp phần thay đổi phương thức tiếp cận thị trường lao động truyền thống, tạo điều kiện thuận lợi cho cả người tìm việc lẫn nhà tuyển dụng trong việc tiếp cận thông tin nhanh chóng và chính xác.

Đặc biệt, trong bối cảnh thị trường lao động ngày càng cạnh tranh và biến động, việc xây dựng một website tìm kiếm và ứng tuyển việc làm chuyên nghiệp không chỉ là xu hướng mà còn trở thành một nhu cầu thiết yếu. Đối với người tìm việc, các nền tảng trực tuyến giúp dễ dàng tra cứu thông tin tuyển dụng, lọc việc làm theo nhiều tiêu chí và nộp hồ sơ trực tuyến một cách thuận tiện. Đối với nhà tuyển dụng, hệ thống tuyển dụng trực tuyến hỗ trợ đăng tin tuyển dụng, quản lý hồ sơ ứng viên và theo dõi hiệu quả tuyển dụng, từ đó nâng cao chất lượng và hiệu suất tuyển dụng nhân sự.

Báo cáo này tập trung trình bày quá trình xây dựng một website tìm kiếm và ứng tuyển việc làm, áp dụng các công nghệ web hiện đại nhằm đảm bảo tính hiệu quả, ổn định và khả năng mở rộng của hệ thống. Cụ thể, Next.js được sử dụng để phát triển giao diện frontend với khả năng tối ưu hiệu năng và trải nghiệm người dùng; Node.js kết hợp Express.js được lựa chọn để xây dựng backend theo mô hình RESTful API; MySQL đảm nhiệm vai trò quản lý và lưu trữ dữ liệu. Bên cạnh đó, mô hình Clean Architecture được áp dụng nhằm tổ chức hệ thống một cách khoa học, tách biệt rõ ràng các lớp chức năng và nâng cao khả năng bảo trì.

Qua việc phân tích yêu cầu, thiết kế cơ sở dữ liệu và triển khai các chức năng cốt lõi của hệ thống, đồ án hướng tới việc xây dựng một nền tảng tuyển dụng trực tuyến đáp ứng nhu cầu thực tế của thị trường lao động hiện nay. Kết quả đạt được không chỉ góp phần cung cấp kiến thức chuyên môn về phát triển ứng dụng web mà còn tạo ra một sản phẩm có tính ứng dụng cao, làm cơ sở cho các nghiên cứu và phát triển nâng cao trong tương lai.

## CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

### 2.1 Next.js

#### 2.1.1 Giới thiệu

Next.js là một framework React dùng để xây dựng các ứng dụng web full-stack. Bạn sử dụng các thành phần React (React Components) để xây dựng giao diện người dùng, và sử dụng Next.js để có thêm các tính năng và tối ưu hóa.

Bên dưới, Next.js cũng trùu tượng hóa và tự động cấu hình các công cụ cần thiết cho React, như bundling (đóng gói), compiling (biên dịch), và nhiều hơn nữa. Điều này cho phép bạn tập trung vào việc xây dựng ứng dụng thay vì tốn thời gian cấu hình.

Dù bạn là một lập trình viên cá nhân hay một thành viên của một đội ngũ lớn, Next.js có thể giúp bạn xây dựng các ứng dụng React tương tác, động và nhanh chóng [5].

#### 2.1.2 Ưu điểm của Next.js

**Kết xuất phía máy chủ (SSR) và Kết xuất tĩnh (SSG):** Next.js hỗ trợ cả SSR và SSG, cho phép tạo ra các trang web với nội dung được tạo ra tại thời điểm yêu cầu hoặc trước đó. Điều này giúp cải thiện thời gian tải trang và trải nghiệm người dùng, đồng thời nâng cao khả năng tìm kiếm trên các công cụ tìm kiếm như Google.

**Tối ưu hóa SEO:** Với khả năng tạo ra nội dung từ phía máy chủ, Next.js giúp cải thiện khả năng tìm kiếm của trang web trên các công cụ tìm kiếm. Việc tối ưu hóa SEO trở nên dễ dàng hơn, đảm bảo rằng trang web của bạn được tìm thấy và xếp hạng cao trên kết quả tìm kiếm.

**Tích hợp sẵn với React và TypeScript:** Next.js tích hợp sẵn với React, một thư viện JavaScript phổ biến được sử dụng rộng rãi trong cộng đồng phát triển. Bạn cũng có thể sử dụng TypeScript để cải thiện tính linh hoạt và dễ bảo trì của mã nguồn của mình.

**Routing đơn giản:** Next.js cung cấp một hệ thống routing đơn giản và mạnh mẽ, giúp bạn tổ chức ứng dụng của mình một cách dễ dàng và hiệu quả. Bạn có thể xác định các tuyến đường và điều hướng dễ dàng thông qua các tệp JavaScript trong thư mục pages.

**Triển khai dễ dàng:** Next.js tích hợp tốt với nhiều dịch vụ phát triển web như Vercel, Netlify và AWS Amplify, giúp việc triển khai và quản lý ứng dụng của bạn trở nên dễ dàng và tiện lợi hơn bao giờ hết.

**Hỗ trợ tốt từ cộng đồng và tài liệu đa dạng:** Next.js có một cộng đồng lớn và sôi động, cung cấp nhiều tài liệu, hướng dẫn và nguồn lực học tập. Bạn có thể dễ dàng tìm kiếm giải pháp cho các vấn đề phát triển và nhận được sự hỗ trợ từ cộng đồng trong quá trình phát triển [3].

### 2.1.3 Nhược điểm của Next.js

Mặc dù Next.js là một framework mạnh mẽ, nhưng nó cũng có một số nhược điểm cần cân nhắc xem có phù hợp với dự án của bạn không

**Chi phí phát triển và bảo trì:** Mặc dù Next.js giúp đơn giản hóa việc xây dựng ứng dụng, nhưng nó vẫn là một framework với cấu trúc riêng. Các developer cần có kiến thức về React và Next.js để xây dựng và bảo trì ứng dụng hiệu quả. So với các thư viện React đơn giản hơn, việc tìm kiếm developer có kinh nghiệm với Next.js có thể tốn nhiều thời gian và chi phí hơn.

**Không có state manager tích hợp sẵn:** Next.js không đi kèm với state manager mặc định. Bạn cần chọn và tích hợp thêm một state manager như Redux, MobX, hoặc Context API của React. Việc lựa chọn và tích hợp thêm state manager có thể làm tăng thêm độ phức tạp của dự án.

**Ít plugin hơn so với các framework khác:** So với các framework lâu đời hơn như Angular hay Vue, Next.js có hệ sinh thái plugin ít phong phú hơn. Mặc dù Next.js hỗ trợ tích hợp với nhiều thư viện của bên thứ ba, bạn có thể gặp khó khăn trong việc tìm kiếm một plugin giải quyết vấn đề cụ thể của mình. Trong một số trường hợp, bạn có thể cần phải xây dựng plugin tùy chỉnh để đáp ứng nhu cầu.

### 2.1.4 Một số đặc điểm cơ bản trong Next.js

#### 2.1.4.1 Routing trong NextJS

**Automatic Routing:** NextJS sẽ tự động tạo các router dựa trên cấu trúc thư mục của chúng ta. Ví dụ, nếu bạn tạo một file có tên là about.js ở thư mục pages. NextJS sẽ tạo router là /about.

**Nested Routing:** Chúng ta có thể tạo các thư mục con để tạo các router lồng nhau. Ví dụ, nếu bạn tạo một folder có tên blog nằm trong folder pages, bên trong folder blog lại có file post.js, đường dẫn sẽ là pages/blog/post.js, thì router mà NextJS tạo ra sẽ là /blog/post.

**Dynamic Routes:** Bạn có thể tạo các router động bằng cách sử dụng cặp dấu [] trong tên file. Ví dụ nếu đường dẫn là pages/blog/[slug].js thì NextJS sẽ tạo ra các router như /blog/blog-dau-tien hoặc /blog/blog-thu-hai. Với slug là một giá trị bất kỳ do bạn truyền vào.

**Link Component:** Để tạo liên kết giữa các trang, bạn sử dụng component Link được cung cấp sẵn bởi NextJS ở thư viện next/link. Sử dụng Link thay cho thẻ a giúp tránh việc tải lại trang và tối ưu hóa hiệu suất.

**Query Parameters:** Bạn có thể truyền dữ liệu giữa các trang sử dụng query parameters trong router bằng cách sử dụng ký tự dấu chấm hỏi ? trong tên file. Ví dụ, pages/product.js có thể có các router như /product?productId=0001 [3].

#### 2.1.4.2 Rendering trong NextJS

**Server-side Rendering (SSR):** Máy chủ xử lý dữ liệu, thực thi JavaScript và gửi HTML đã kết xuất sẵn về trình duyệt. Điều này giúp tải nhanh hơn và tối ưu SEO vì nội dung đã được hiển thị trước khi đến tay khách hàng.

**Client-side Rendering (CSR):** Máy chủ chỉ gửi một trang HTML cơ bản và các tệp JavaScript. Trình duyệt sau đó sẽ thực thi JavaScript, lấy dữ liệu từ API và tự kết xuất nội dung, phù hợp cho các ứng dụng web động và nhiều tương tác.

**Static Site Generation (SSG):** Là một phương pháp mà NextJS cung cấp sẵn cho chúng ta, cho phép bạn tạo các trang tĩnh và lưu chúng xuống dưới dạng file html tĩnh. Điều này giúp cải thiện hiệu suất tải trang và cung cấp trải nghiệm người dùng tốt hơn vì nội dung được lấy từ file html và hiển thị ngay lập tức mà không cần đợi việc tải về từ phía server [3].

#### 2.1.4.3 Styling trong NextJS

**CSS Modules:** Để style cho ứng dụng NextJS, cách dễ nhất là bạn có thể tạo các file CSS/SCSS riêng lẻ cho từng component hoặc sử dụng file chung cho toàn dự án.

**CSS Frameworks:** NextJS cũng hỗ trợ sử dụng cùng các CSS framework như TailwindCSS, Bootstrap hoặc MaterialUI [3].

## 2.2 ExpressJS

### 2.2.1 Giới thiệu

ExpressJS là một framework mạnh mẽ và phổ biến được xây dựng trên nền tảng Node.js. Nó giúp đơn giản hóa quá trình phát triển ứng dụng web bằng cách cung cấp các công cụ và thư viện cần thiết để xử lý yêu cầu và phản hồi HTTP một cách dễ dàng [8].

### 2.2.2 Ưu điểm của ExpressJS

**Đơn giản và dễ sử dụng:** ExpressJS có cú pháp đơn giản và dễ hiểu, giúp lập trình viên dễ dàng nắm bắt và triển khai các tính năng.

**Linh hoạt:** ExpressJS không áp đặt một cấu trúc cụ thể, cho phép lập trình viên tự do tùy chỉnh và xây dựng ứng dụng theo ý muốn.

**Hỗ trợ middleware:** ExpressJS cung cấp hệ thống middleware mạnh mẽ, cho phép thực hiện các chức năng như xác thực, ghi log, nén dữ liệu và xử lý lỗi một cách linh hoạt và dễ dàng.

**Hiệu suất cao:** ExpressJS được xây dựng trên Node.js, nền tảng có hiệu suất cao, cho phép xử lý nhanh chóng các yêu cầu web đồng thời và có khả năng mở rộng tốt [8].

### 2.2.3 Nhược điểm của ExpressJS

Bên cạnh rất nhiều những ưu điểm nổi bật kể trên, ExpressJS cũng tồn tại một số hạn chế sau:

**Thiếu cấu trúc:** Do ExpressJS không áp đặt một cấu trúc nghiêm ngặt, việc tổ chức dự án và quản lý mã nguồn có thể trở nên khó khăn, đặc biệt khi ứng dụng phát triển lớn và phức tạp.

**Khả năng mở rộng:** Khi ứng dụng phát triển lớn và phức tạp, việc quản lý mã nguồn và mở rộng có thể trở nên khó khăn với ExpressJS. Cần có sự kiểm soát cẩn thận để tránh sự phức tạp và rối rắm trong việc quản lý các module và tương tác giữa chúng.

**Cộng đồng hỗ trợ:** Mặc dù ExpressJS có một cộng đồng lớn và đầy đủ tài liệu, tuy nhiên, không đạt được mức độ hỗ trợ như các framework web khác như Angular hoặc React [8].

#### 2.2.4 Một số tính năng của ExpressJS

**Phát triển máy chủ nhanh hơn:** ExpressJS tối ưu hóa cú pháp và cung cấp các phương thức và hàm tiện ích để xử lý các tác vụ phổ biến trong lập trình web. Nhờ đó, bạn có thể viết code ngắn gọn và tối giản hóa quy trình phát triển.

**Định tuyến (Routing):** ExpressJS cung cấp một hệ thống định tuyến mạnh mẽ, cho phép bạn xác định các tuyến đường (routes) để xử lý yêu cầu từ người dùng và phản hồi tương ứng. Điều này giúp tổ chức và quản lý các thành phần của ứng dụng một cách dễ dàng.

**Middleware:** ExpressJS hỗ trợ middleware, cho phép bạn thêm các chức năng trung gian vào quy trình xử lý yêu cầu và phản hồi. Middleware giúp xác thực người dùng, ghi log, xử lý lỗi, nén dữ liệu và thực hiện nhiều tác vụ khác một cách linh hoạt.

**Cấu hình môi trường:** ExpressJS cung cấp một cách để cấu hình môi trường phát triển và môi trường sản xuất. Bạn có thể thiết lập các biến môi trường, cấu hình định dạng và quy tắc, tùy chỉnh ứng dụng của mình theo các môi trường khác nhau.

**Xử lý lỗi:** ExpressJS cung cấp cơ chế xử lý lỗi cho phép bạn kiểm soát và xử lý các lỗi xảy ra trong quá trình xử lý yêu cầu. Bạn có thể tạo ra các middleware để xử lý lỗi và phản hồi với các thông báo lỗi tùy chỉnh.

### 2.3 Node.js

#### 2.3.1 Giới thiệu

Node.js là môi trường thực thi JavaScript mã nguồn mở và đa nền tảng, cho phép xây dựng các ứng dụng phía máy chủ hiệu năng cao. Node.js sử dụng bộ máy JavaScript V8 để biên dịch và thực thi mã nguồn, giúp tối ưu tốc độ xử lý.

Node.js hoạt động theo mô hình bát đồng bộ, hướng sự kiện và không chẵn luồng. Thay vì tạo một luồng riêng cho mỗi yêu cầu, Node.js xử lý các thao tác vào/ra như truy cập cơ sở dữ liệu, hệ thống tập tin hoặc mạng theo cơ chế không đồng bộ, cho phép tiếp tục xử lý các tác vụ khác trong khi chờ phản hồi.

Nhờ đặc điểm này, Node.js có khả năng xử lý số lượng lớn kết nối đồng thời với mức tiêu thụ tài nguyên thấp, phù hợp cho các hệ thống web có yêu cầu cao về hiệu năng và khả năng mở rộng. Ngoài ra, việc sử dụng JavaScript cho cả frontend và backend giúp đơn giản hóa quá trình phát triển và bảo trì hệ thống [6].

### 2.3.2 Ưu điểm của Node.js

**Hiệu suất cao:** Nodejs chạy đơn luồng, sử dụng V8 Engine, giúp ứng dụng đảm bảo tốc độ khi có nhiều requests.

**Xử lý bất đồng bộ và I/O hướng sự kiện:** Khả năng xử lý I/O bất đồng bộ, giúp Nodejs có thể xử lý nhiều tasks, mà không cần phải chờ kết quả của task trước đó.

**Phát triển ứng dụng:** Có thể sử dụng để phát triển ứng dụng ở cả phía client và server.

**Module đa dạng:** Nodejs sở hữu một cộng đồng duy trì, phát triển modules, thư viện giúp cho việc phát triển ứng dụng nhanh chóng.

**Stream và xử lý file lớn:** Nodejs hỗ trợ streaming, cho phép xử lý các file có kích thước lớn không tồn tại nguyễn.

**Phù hợp với ứng dụng real time:** Do Nodejs xử lý bất đồng bộ, thích hợp với các ứng dụng real time như: chat applications, streaming services,...

### 2.3.3 Nhược điểm của Node.js

**Không phù hợp với các tác vụ CPU-Intensive:** Node.js hoạt động trên một luồng đơn, giúp xử lý I/O không đồng bộ nhưng lại gặp khó khăn với các tác vụ tính toán nặng như xử lý ảnh, video hay các thuật toán phức tạp.

**Callback Hell và khó khăn trong xử lý bất đồng bộ:** Việc lồng nhau quá nhiều callback dẫn đến code khó đọc và bảo trì.

**Thiếu tính nhất quán giữa các phiên bản Node.js:** Một số package hoặc module có thể không tương thích với các phiên bản Node.js mới do các thay đổi về API.

**Rủi ro bảo mật và quản lý thư viện npm:** Hàng triệu package trên npm mang lại lợi ích nhưng cũng tiềm ẩn nguy cơ bảo mật nếu sử dụng các module không được kiểm duyệt kỹ lưỡng.

### 2.3.4 Một số đặc điểm cơ bản của Node.js

#### 2.3.4.1 Mô hình bất đồng bộ (Asynchronous)

**Asynchronous Processing:** Node.js xử lý các tác vụ theo cơ chế bất đồng bộ, cho phép thực hiện các thao tác vào/ra (I/O) như truy vấn cơ sở dữ liệu, đọc ghi tệp tin hoặc gọi API mà không làm chận luồng chính. Nhờ đó, hệ thống có thể tiếp tục xử lý các yêu cầu khác trong khi chờ phản hồi, giúp nâng cao hiệu suất và khả năng đáp ứng.

#### 2.3.4.2 Mô hình hướng sự kiện (Event-driven)

**Event-driven Architecture:** Node.js hoạt động dựa trên cơ chế hướng sự kiện, trong đó mỗi hành động của người dùng hoặc hệ thống sẽ phát sinh một sự kiện tương ứng. Các sự kiện này được quản lý thông qua Event Loop, giúp ứng dụng phản hồi nhanh chóng và xử lý hiệu quả nhiều kết nối đồng thời.

#### 2.3.4.3 Mô hình đơn luồng (Single-threaded)

**Single-threaded Model:** Node.js sử dụng một luồng chính để xử lý các yêu cầu, kết hợp với cơ chế bất đồng bộ nhằm tránh tình trạng nghẽn tài nguyên. Mô hình này giúp giảm chi phí quản lý luồng, hạn chế lỗi liên quan đến đồng bộ dữ liệu và phù hợp với các ứng dụng web có lượng truy cập lớn.

#### 2.3.4.4 Khả năng xây dựng API hiệu quả

**RESTful API Support:** Node.js hỗ trợ xây dựng các API RESTful một cách linh hoạt, dễ dàng tích hợp với các framework như Express.js. Điều này giúp hệ thống backend có cấu trúc rõ ràng, dễ mở rộng và thuận tiện trong việc kết nối với các ứng dụng frontend.

#### 2.3.4.5 Hệ sinh thái thư viện phong phú

**npm Ecosystem:** Node.js đi kèm với npm (Node Package Manager), cung cấp kho thư viện lớn với nhiều gói hỗ trợ xác thực, bảo mật, xử lý dữ liệu và kết nối cơ sở dữ liệu. Nhờ đó, quá trình phát triển ứng dụng được rút ngắn và tối ưu hơn.

### 2.4 Clean Architecture

#### 2.4.1 Giới thiệu

Clean Architecture là một phương pháp thiết kế phần mềm do Robert C. Martin - một chuyên gia phần mềm được biết đến với biệt danh "Uncle Bob", đề xuất. Robert

C. Martin đã viết về Clean Architecture trong cuốn sách "Clean Architecture: A Craftsman's Guide to Software Structure and Design". Trong sách này, ông giải thích chi tiết về nguyên lý và lợi ích của Clean Architecture, cùng với các ví dụ và hướng dẫn cụ thể để áp dụng mô hình này vào các dự án phần mềm thực tế.

Mục tiêu chính của Clean Architecture là tách biệt các thành phần và xây dựng kiến trúc linh hoạt, bền vững và dễ bảo trì. Mô hình này tập trung vào việc tách biệt logic kinh doanh khỏi các yếu tố kỹ thuật, đảm bảo rằng thay đổi trong một phần của hệ thống không ảnh hưởng đến các phần khác. Đồng thời, Clean Architecture cũng đảm bảo mã nguồn dễ đọc, dễ kiểm thử và dễ hiểu. Với đóng góp của Robert C. Martin, Clean Architecture đã trở thành một khung kiến trúc phổ biến và được áp dụng rộng rãi trong ngành công nghiệp phát triển phần mềm, giúp xây dựng các hệ thống chất lượng và dễ quản lý [4].

#### **2.4.2 Ưu điểm của Clean Architecture**

**Dễ bảo trì:** Clean Architecture tách biệt các thành phần, làm cho việc bảo trì và sửa lỗi trở nên dễ dàng hơn. Thay đổi trong một phần của hệ thống không ảnh hưởng đến các phần khác.

**Dễ mở rộng:** Kiến trúc Clean Architecture cho phép mở rộng hệ thống một cách linh hoạt. Các thành phần có thể được thêm vào hoặc thay đổi mà không làm ảnh hưởng đến các thành phần khác.

**Độc lập công nghệ:** Clean Architecture giúp giữ cho mã nguồn không phụ thuộc vào các công nghệ cụ thể. Điều này cho phép dễ dàng chuyển đổi và cập nhật công nghệ mà không ảnh hưởng đến kiến trúc cốt lõi của hệ thống. Độc lập định dạng giao diện người dùng: Clean Architecture cho phép thay đổi định dạng giao diện người dùng mà không ảnh hưởng đến logic kinh doanh của hệ thống.

**Kiểm thử dễ dàng:** Kiến trúc Clean Architecture tách biệt logic kinh doanh, làm cho việc kiểm thử trở nên dễ dàng hơn và giúp đảm bảo tính ổn định và chính xác của hệ thống [4].

#### **2.4.3 Nhược điểm của Clean Architecture**

**Công kẽm và phức tạp:** Điều dễ thấy nhất là Clean Architecture không hề dễ sử dụng, phải viết nhiều lớp (class/object) hơn. Trong trường hợp ứng dụng của bạn

quá đơn giản, ít tính năng, vòng đời ngắn thì chọn lựa kiến trúc này có thể mang lại những rắc rối không cần thiết.

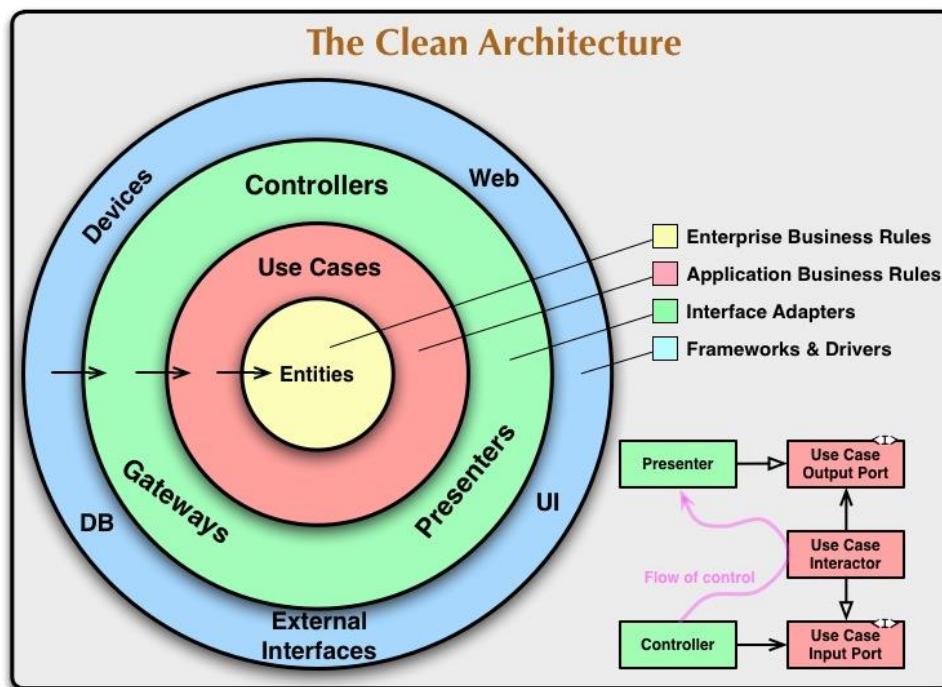
**Tính trừu tượng cao:** Vấn đề này gọi là indirect. Trừu tượng càng cao thì tiện cho các developers nhưng sẽ gây ảnh hưởng không nhỏ tới tốc độ thực thi (performance). Ngoài ra cũng không thể code nhanh, với vã "mì ăn liền" được mà phải tạo đủ các Interfaces.

**Khó tuyển người:** Sử dụng Clean Architecture sẽ cần tuyển dụng developer thấu hiểu về kiến trúc này. Nguyên tắc Dependency Inversion rất dễ bị xâm phạm vì sự hạn chế kiến thức, sự bất cẩn hoặc vì thời gian cần triển khai tính năng quá ít [2].

#### 2.4.4 Một số đặc điểm cơ bản trong Clean Architecture

Clean Architecture loại bỏ sự lệ thuộc giữa các đối tượng cũng như các layer trong ứng dụng. Nguyên lý này kế thừa và phát triển dựa trên Dependency Inversion - nguyên lý nổi tiếng trong SOLID.

Trong kiến trúc Clean Architecture bao gồm 4 layer được đại diện thông qua các vòng tròn đồng tâm. Các vòng tròn ở trong sẽ không hề biết gì về các vòng tròn bên ngoài. Nguyên tắc "hướng tâm" này được minh họa như sau:



Hình 2.1 Clean Architecture

Từ trong ra ngoài Clean Architecture sẽ bao gồm: Entities, Use Cases, Interface Adapters và Frameworks & Drivers.

Về cơ bản các layer này sẽ làm việc qua các trùu tượng của nhau (interfaces).

### **Entities**

Entities là layer trong cùng, cũng là layer quan trọng nhất. Entity chính là các thực thể hay từng đối tượng cụ thể và các rule business logic của nó. Trong OOP, đây chính là Object cùng với các method và properties tuân thủ nguyên tắc Encapsulation - chỉ trong Object mới có thể thay đổi trạng thái (State) của chính nó.

Ví dụ: Trong object Person thì thuộc tính age không thể bé hơn 1. Nếu cần thay đổi age, chúng ta phải viết hàm public setAge, hàm này cũng chịu trách nhiệm check điều kiện liên quan tới age.

Các business logic của layer Entities sẽ không quan tâm hay lệ thuộc vào các business logic ở các layer bên ngoài như Use Cases. Giả sử với trường hợp người dùng phải từ 18 tuổi trở lên mới được phép tạo tài khoản thì rule thuộc tính Age trong Entities vẫn không đổi.

### **Use Cases**

Use Cases là layer chứa các business logic ở cấp độ cụ thể từng Use Case (hay application).

VD: Use Case đăng ký tài khoản (tạo mới một Person/Account) sẽ cần tổ hợp một hoặc nhiều Entities tùy vào độ phức tạp của Use Case.

Các business logic của Use Case đương nhiên cũng sẽ không quan tâm và lệ thuộc vào việc dữ liệu đến từ đâu, dùng các thư viện nào làm adapter, dữ liệu thể hiện thế nào,... Vì đây là nhiệm vụ của layer Interface Adapters.

### **Interface Adapters**

Interface Adapters chính là layer phụ trách việc chuyển đổi các format dữ liệu để phù hợp với từng Use Case và Entities. Các format dữ liệu này có thể dùng cho cả bên trong hoặc ngoài ứng dụng.

VD: Thông tin người dùng sẽ có một số thông tin rất nhạy cảm như Email, Phone, Address. Không phải lúc nào dữ liệu cũng về đầy đủ để phục vụ GUI (Web, App). Tương tự với tùy vào hệ thống Database mà các adapter phải format dữ liệu hợp lý.

Như vậy dữ liệu đầu vào và ra ở tầng Interface Adapter chỉ cần đủ và hợp lý. Nó sẽ không quan tâm việc dữ liệu sẽ được hiển thị cụ thể như thế nào cũng như được thu thập như thế nào. Vì đó là nhiệm vụ của tầng Frameworks & Drivers.

### **Frameworks & Drivers**

Frameworkd & Drivers là tầng ngoài cùng, tổ hợp các công cụ cụ thể phục vụ cho từng nhu cầu của end user như: thiết bị (devices), web, application, databases,... Trong kiến trúc Clean Architecture thì ở tầng này là "nhẹ" nhất vì chúng ta không cần phải viết quá nhiều code.

Trên thực tế thì đây là nơi "biết tất cả" cụ thể các tầng là gì thông qua việc chịu trách nhiệm khởi tạo các objects cho các tầng bên trong (hay còn gọi là Setup Dependencies)

Để các layer trong Clean Architecture có thể làm việc được nhưng lại độc lập với nhau thì chúng sẽ dùng các Interfaces [2].

## **2.5 RESTful API**

### **2.5.1 Giới thiệu**

REST là viết tắt của REpresentational State Transfer (Chuyển đổi trạng thái đại diện) và là một phong cách kiến trúc dành cho các hệ thống siêu phương tiện phân tán. Roy Fielding đã giới thiệu REST lần đầu tiên vào năm 2000 trong luận án nổi tiếng của mình. Kể từ đó, REST đã trở thành một trong những cách tiếp cận được sử dụng rộng rãi nhất để xây dựng các API (Application Programming Interfaces) dựa trên web.

REST không phải là một giao thức hay tiêu chuẩn, mà là một phong cách kiến trúc. Trong quá trình phát triển, các nhà phát triển API có thể triển khai REST theo nhiều cách khác nhau.

Giống như các phong cách kiến trúc khác, REST cũng có các nguyên tắc và ràng buộc hướng dẫn của riêng nó. Các nguyên tắc này cần được đáp ứng nếu giao diện dịch vụ muốn được gọi là RESTful.

API Web (hoặc dịch vụ web) tuân theo kiểu kiến trúc REST được gọi là REST API (hoặc RESTful API) [7].

### 2.5.2 Nguyên tắc cơ bản của RESTful API

#### Giao diện đồng nhất (Uniform Interface)

Bằng cách áp dụng nguyên tắc chung cho giao diện của các thành phần, chúng ta có thể đơn giản hóa kiến trúc hệ thống tổng thể và cải thiện khả năng quan sát các tương tác. Một số ràng buộc kiến trúc hỗ trợ đạt được giao diện đồng nhất và hướng dẫn hành vi của các thành phần.

Bốn ràng buộc sau đây giúp đạt được giao diện đồng nhất của REST:

- Xác định tài nguyên – Giao diện phải định danh duy nhất từng tài nguyên tham gia vào tương tác giữa client và server.
- Thao tác tài nguyên thông qua các đại diện – Tài nguyên nên có các đại diện đồng nhất trong phản hồi của server. Người dùng API nên sử dụng các đại diện này để sửa đổi trạng thái tài nguyên trên server.
- Thông điệp tự mô tả – Mỗi đại diện tài nguyên nên mang đủ thông tin để mô tả cách xử lý thông điệp. Đồng thời, nó cũng nên cung cấp thông tin về các hành động bổ sung mà client có thể thực hiện trên tài nguyên.
- Hypermedia là động cơ của trạng thái ứng dụng (HATEOAS) – Client chỉ cần biết URI ban đầu của ứng dụng. Từ đó, ứng dụng client nên tự động điều hướng các tài nguyên và tương tác khác thông qua các siêu liên kết.

Nói cách đơn giản hơn, REST định nghĩa một giao diện nhất quán và đồng nhất để tương tác giữa client và server. Ví dụ, các API REST dựa trên HTTP sử dụng các phương thức HTTP chuẩn (GET, POST, PUT, DELETE, v.v.) và URIs (Uniform Resource Identifiers) để định danh tài nguyên.

#### Mô hình Client-Server

Mô hình thiết kế client-server bắt buộc tách biệt các mối quan tâm, giúp các thành phần client và server phát triển độc lập.

Bằng cách tách rời mỗi quan tâm về giao diện người dùng (client) khỏi mỗi quan tâm về lưu trữ dữ liệu (server), chúng ta cải thiện khả năng di chuyển giao diện

người dùng trên nhiều nền tảng khác nhau và nâng cao khả năng mở rộng bằng cách đơn giản hóa các thành phần server.

Khi client và server phát triển, cần đảm bảo rằng giao diện/hợp đồng giữa chúng không bị phá vỡ.

### **Không trạng thái (Stateless)**

Tính không trạng thái yêu cầu mỗi yêu cầu từ client đến server phải chứa tất cả thông tin cần thiết để hiểu và hoàn thành yêu cầu. Server không thể tận dụng bất kỳ thông tin ngữ cảnh nào được lưu trữ trước đó trên server. Vì lý do này, ứng dụng client phải hoàn toàn quản lý trạng thái phiên.

### **Có thể lưu bộ nhớ đệm (Cacheable)**

Ràng buộc về khả năng lưu bộ nhớ đệm yêu cầu một phản hồi phải được gắn nhãn rõ ràng là có thể lưu bộ nhớ đệm hoặc không. Nếu phản hồi có thể lưu bộ nhớ đệm, ứng dụng client có quyền tái sử dụng dữ liệu phản hồi cho các yêu cầu tương đương trong một khoảng thời gian được chỉ định.

### **Hệ thống phân lớp (Layered System)**

Phong cách hệ thống phân lớp cho phép kiến trúc được tổ chức thành các lớp phân cấp bằng cách ràng buộc hành vi của các thành phần. Trong một hệ thống phân lớp, mỗi thành phần không thể nhìn xa hơn lớp ngay lập tức mà nó đang tương tác. Ví dụ đơn giản về hệ thống phân lớp là mô hình MVC. Mô hình MVC cho phép tách biệt rõ ràng các mối quan tâm, giúp phát triển, duy trì và mở rộng ứng dụng dễ dàng hơn.

### **Mã theo yêu cầu (Code on Demand - Tùy Chọn)**

REST cũng cho phép mở rộng chức năng client bằng cách tải xuống và thực thi mã dưới dạng applet hoặc script. Mã tải xuống này giúp đơn giản hóa client bằng cách giảm số lượng tính năng cần được triển khai trước. Server có thể cung cấp một phần tính năng dưới dạng mã, và client chỉ cần thực thi mã đó.

#### **2.5.3 Ưu điểm của RESTful API**

**Khả năng thay đổi quy mô:** Các hệ thống triển khai API REST có thể thay đổi quy mô một cách hiệu quả vì REST tối ưu hóa các tương tác giữa client và máy chủ. Tình trạng phi trạng thái loại bỏ tải của máy chủ vì máy chủ không phải giữ lại thông

tin yêu cầu của client trong quá khứ. Việc lưu bộ nhớ đệm được quản lý tốt sẽ loại bỏ một phần hoặc hoàn toàn một số tương tác giữa client và máy chủ. Tất cả các tính năng này hỗ trợ khả năng thay đổi quy mô mà không gây ra tắc nghẽn giao tiếp làm giảm hiệu suất.

**Sự linh hoạt:** Các dịch vụ web RESTful hỗ trợ phân tách hoàn toàn giữa client và máy chủ. Các dịch vụ này đơn giản hóa và tách riêng các thành phần máy chủ khác nhau để mỗi phần có thể phát triển độc lập. Các thay đổi ở nền tảng hoặc công nghệ tại ứng dụng máy chủ không ảnh hưởng đến ứng dụng client. Khả năng phân lớp các chức năng ứng dụng làm tăng tính linh hoạt hơn nữa. Ví dụ: các nhà phát triển có thể thực hiện các thay đổi đối với lớp cơ sở dữ liệu mà không cần viết lại logic ứng dụng.

**Sự độc lập:** Các API REST không phụ thuộc vào công nghệ được sử dụng. Bạn có thể viết cả ứng dụng client và máy chủ bằng nhiều ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến thiết kế API. Bạn cũng có thể thay đổi công nghệ cơ sở ở hai phía mà không ảnh hưởng đến giao tiếp.

#### 2.5.4 Nhược điểm RESTful API

**Bảo mật:** RESTful API cung cấp cho các ứng dụng một giao diện lập trình để sử dụng, tuy nhiên điều này cũng đồng nghĩa với việc có thể gây ra những vấn đề về bảo mật. Các nhà phát triển cần phải đảm bảo rằng RESTful API của họ được bảo vệ an toàn để tránh các cuộc tấn công từ bên ngoài và tình trạng lỗ hổng bảo mật.

**Khó khăn trong việc quản lý phiên:** RESTful API không hỗ trợ quản lý phiên, điều này gây ra khó khăn trong việc xác thực và quản lý người dùng. Khi sử dụng, các nhà phát triển cần phải có chính sách xác thực phù hợp để giữ cho thông tin của người dùng được bảo vệ an toàn.

## 2.6 MySQL

### 2.6.1 Giới thiệu

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System - RDBMS) mã nguồn mở, sử dụng ngôn ngữ truy vấn cấu trúc (SQL) để quản lý và thao tác dữ liệu. MySQL tổ chức và lưu trữ dữ liệu dưới dạng các bảng, trong đó các bảng có thể liên kết với nhau thông qua các khóa (keys).

MySQL được sử dụng rộng rãi trong việc phát triển các ứng dụng web. Nhiều công ty lớn như: Facebook, Twitter, và YouTube sử dụng MySQL để quản lý dữ liệu.

MySQL được tạo ra bởi ba nhà phát triển: Michael Widenius, David Axmark và Allan Larsson vào năm 1995. Họ muốn tạo ra một hệ quản trị cơ sở dữ liệu nhanh, đáng tin cậy và dễ sử dụng cho các dự án của mình. Năm 2008, công ty Sun Microsystems mua lại MySQL AB, công ty phát triển MySQL, với giá 1 tỷ USD. Sau đó, vào năm 2010, Oracle Corporation mua lại Sun Microsystems, và từ đó, MySQL trở thành một phần của Oracle [1].

### 2.6.2 Ưu điểm của MySQL

MySQL là phần mềm mã nguồn mở, cho phép người dùng sử dụng và tùy chỉnh mà không cần phải trả phí.

Được thiết kế để xử lý dữ liệu nhanh chóng và hiệu quả, MySQL phù hợp với các ứng dụng web đòi hỏi tốc độ cao và khả năng phản hồi nhanh.

Có nhiều tài liệu, hướng dẫn cú pháp SQL phù hợp với cả người mới bắt đầu.

Cung cấp nhiều tính năng bảo mật như xác thực người dùng, mã hóa dữ liệu và quản lý quyền truy cập chi tiết, giúp bảo vệ dữ liệu quan trọng.

MySQL có thể xử lý từ các cơ sở dữ liệu nhỏ đến các hệ thống lớn với hàng triệu bản ghi, dễ dàng mở rộng khi nhu cầu tăng lên mà không cần thay đổi cấu trúc hệ thống.

Hỗ trợ các tính năng transaction đảm bảo tính toàn vẹn và nhất quán của dữ liệu, giúp quản lý các thao tác dữ liệu phức tạp [1].

### 2.6.3 Nhược điểm MySQL

**Hạn chế dung lượng:** Trong trường hợp số lượng bản ghi ngày càng gia tăng, quá trình truy xuất dữ liệu sẽ trở nên khó khăn. Để giải quyết vấn đề này, bạn có thể áp dụng nhiều biện pháp như tạo cache MySQL hoặc phân tải cơ sở dữ liệu ra nhiều máy chủ.

**Độ tin cậy thấp:** Một nhược điểm khác của MySQL là trong quá trình thực hiện các chức năng cụ thể như kiểm toán, giao dịch và quản lý tài liệu tham khảo, MySQL có thể trở nên kém tin cậy hơn một số hệ quản trị cơ sở dữ liệu quan hệ khác.

**Giới hạn chức năng:** MySQL không thiết kế để thực hiện toàn bộ các chức năng và nó đi kèm với những hạn chế về chức năng mà một số ứng dụng có thể cần.

## CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

### 3.1 Mô tả bài toán

Xây dựng một hệ thống website tìm kiếm và ứng tuyển việc làm trực tuyến nhằm hỗ trợ kết nối hiệu quả giữa người tìm việc và nhà tuyển dụng. Hệ thống giúp khắc phục các hạn chế của hình thức tuyển dụng truyền thống như thông tin việc làm phân tán, quy trình ứng tuyển thủ công, thiếu công cụ quản lý hồ sơ và chưa tối ưu trải nghiệm người dùng. Đối với người tìm việc, hệ thống cho phép tra cứu, tìm kiếm và lọc các công việc phù hợp theo nhiều tiêu chí như ngành nghề, địa điểm, mức lương, loại hình công việc, kinh nghiệm và trình độ học vấn. Người dùng có thể quản lý thông tin cá nhân, lưu các công việc quan tâm, theo dõi nhà tuyển dụng và thực hiện ứng tuyển trực tuyến thông qua hệ thống. Đối với nhà tuyển dụng, hệ thống hỗ trợ đăng tin tuyển dụng, quản lý danh sách công việc, tiếp nhận và xử lý hồ sơ ứng tuyển của ứng viên, đồng thời theo dõi trạng thái ứng tuyển một cách trực quan. Ngoài ra, hệ thống cần cung cấp cơ chế thông báo nhằm tăng cường khả năng tương tác giữa ứng viên và nhà tuyển dụng trong quá trình tuyển dụng. Đồng thời cung cấp tính năng đánh giá hồ sơ xin việc (CV) với mô hình ngôn ngữ lớn (LLM) để người tìm việc hoặc nhà tuyển dụng có thể sử dụng để đánh giá CV của người tìm việc.

### 3.2 Đặc tả yêu cầu hệ thống

#### 3.2.1 Yêu cầu chức năng

##### Chức năng của người tìm việc (ứng viên)

**Tìm kiếm và lọc việc làm:** Người tìm việc có thể tìm kiếm các vị trí tuyển dụng dựa trên từ khóa tên công việc. Hệ thống hỗ trợ chức năng lọc việc làm theo nhiều tiêu chí như ngành nghề, địa điểm làm việc, mức lương, loại hình công việc, kinh nghiệm và trình độ học vấn, giúp người dùng nhanh chóng tiếp cận các cơ hội việc làm phù hợp với nhu cầu cá nhân.

**Xem chi tiết việc làm:** Mỗi tin tuyển dụng có một trang chi tiết cung cấp đầy đủ thông tin bao gồm tên công việc, mô tả công việc, yêu cầu tuyển dụng, mức lương, hình thức làm việc, thông tin doanh nghiệp và thời gian đăng tuyển, giúp ứng viên có cơ sở đánh giá trước khi ứng tuyển.

**Quản lý hồ sơ cá nhân:** Người tìm việc có thể tạo và cập nhật thông tin cá nhân như họ tên, thông tin liên hệ, giới thiệu bản thân, liên kết mạng xã hội và ảnh đại diện. Các thông tin này được sử dụng trong quá trình ứng tuyển việc làm.

**Ứng tuyển việc làm:** Hệ thống cho phép người tìm việc nộp hồ sơ ứng tuyển trực tuyến bằng cách gửi CV và thư ứng tuyển cho nhà tuyển dụng. Trạng thái ứng tuyển được lưu trữ và hiển thị để người dùng theo dõi quá trình xử lý hồ sơ.

**Lưu việc làm và theo dõi nhà tuyển dụng:** Người tìm việc có thể lưu các công việc quan tâm để xem lại sau và theo dõi nhà tuyển dụng nhằm nhận thông báo khi có tin tuyển dụng mới.

**Đánh giá CV:** Người tìm việc có thể sử dụng tính năng đánh giá CV với AI để có thể nhận những góp ý cải thiện, những kỹ năng, tiêu chí đã đáp ứng với công việc dự định ứng tuyển từ AI.

### Chức năng của nhà tuyển dụng

**Quản lý thông tin doanh nghiệp:** Nhà tuyển dụng có thể tạo và cập nhật hồ sơ doanh nghiệp bao gồm tên công ty, mô tả, quy mô, thông tin liên hệ, website và hình ảnh đại diện nhằm tăng mức độ tin cậy và thu hút ứng viên.

**Đăng và quản lý tin tuyển dụng:** Hệ thống cho phép nhà tuyển dụng đăng mới, chỉnh sửa, ẩn hoặc xóa tin tuyển dụng. Mỗi tin tuyển dụng bao gồm các thông tin cần thiết như mô tả công việc, yêu cầu, mức lương và địa điểm làm việc.

**Quản lý hồ sơ ứng tuyển:** Nhà tuyển dụng có thể xem danh sách ứng viên đã ứng tuyển cho từng vị trí, xem chi tiết hồ sơ, cập nhật trạng thái ứng tuyển và phản hồi kết quả cho ứng viên.

**Đánh giá CV:** Nhà tuyển dụng có thể sử dụng tính năng để đánh giá CV của người tìm việc với vị trí công việc mà họ dự định ứng tuyển với sự hỗ trợ của AI.

### Chức năng thông báo

**Gửi và nhận thông báo:** Hệ thống hỗ trợ gửi thông báo đến người tìm việc và nhà tuyển dụng khi có các sự kiện quan trọng như nộp hồ sơ ứng tuyển, phản hồi từ nhà tuyển dụng hoặc theo dõi doanh nghiệp, giúp tăng cường khả năng tương tác và cập nhật thông tin kịp thời.

### 3.2.2 Yêu cầu phi chức năng

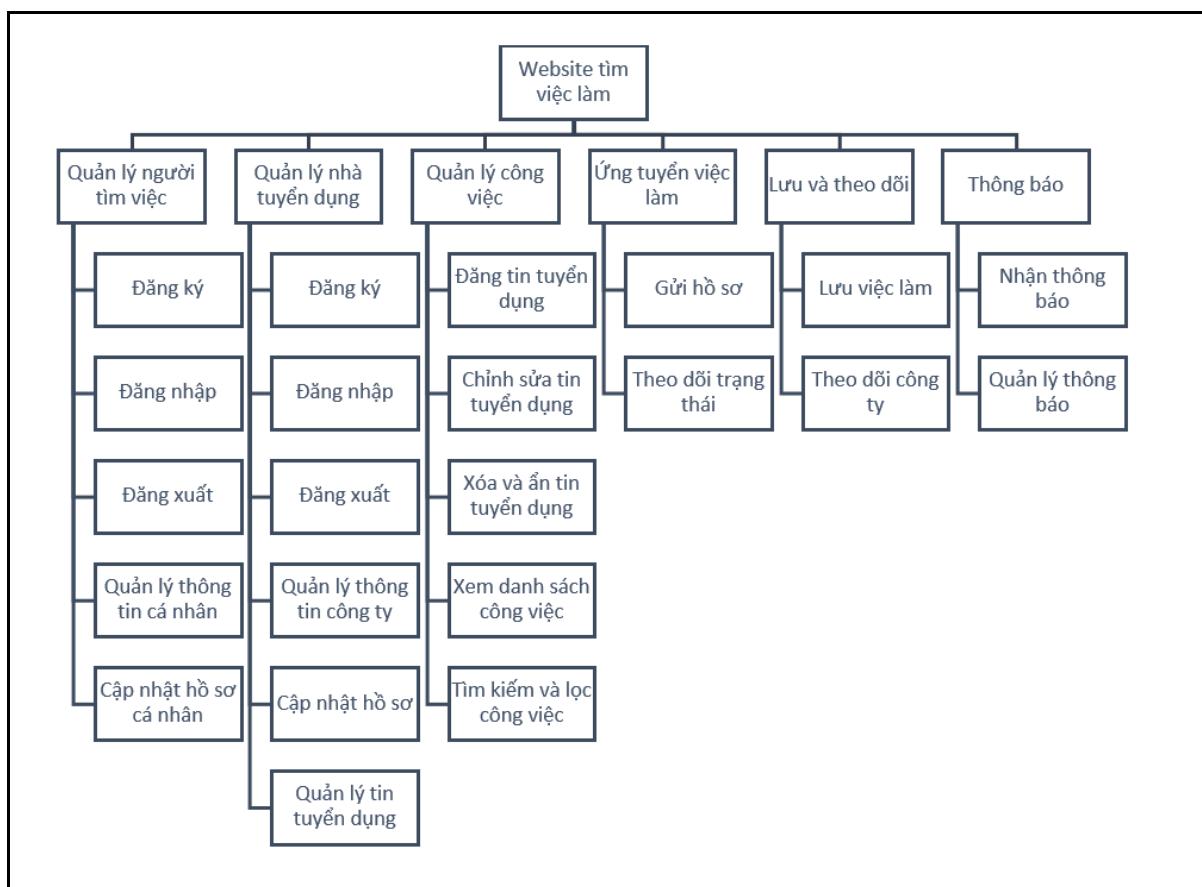
**Hiệu năng:** Hệ thống được thiết kế và tối ưu nhằm đảm bảo tốc độ tải trang nhanh, thời gian phản hồi ngắn và khả năng xử lý đồng thời nhiều yêu cầu từ người dùng.

**Giao diện:** Giao diện hệ thống được xây dựng theo hướng trực quan, đơn giản và dễ sử dụng, phù hợp với nhiều nhóm đối tượng người dùng.

**Bảo mật:** Bảo mật thông tin người dùng, đặc biệt là thông tin cá nhân và dữ liệu xác thực.

**Tương thích:** Đảm bảo rằng trang web hoạt động mượt mà trên nhiều trình duyệt và thiết bị khác nhau, đáp ứng nhanh chóng đến nhu cầu đa dạng của người dùng.

## 3.3 Sơ đồ phân cấp chức năng



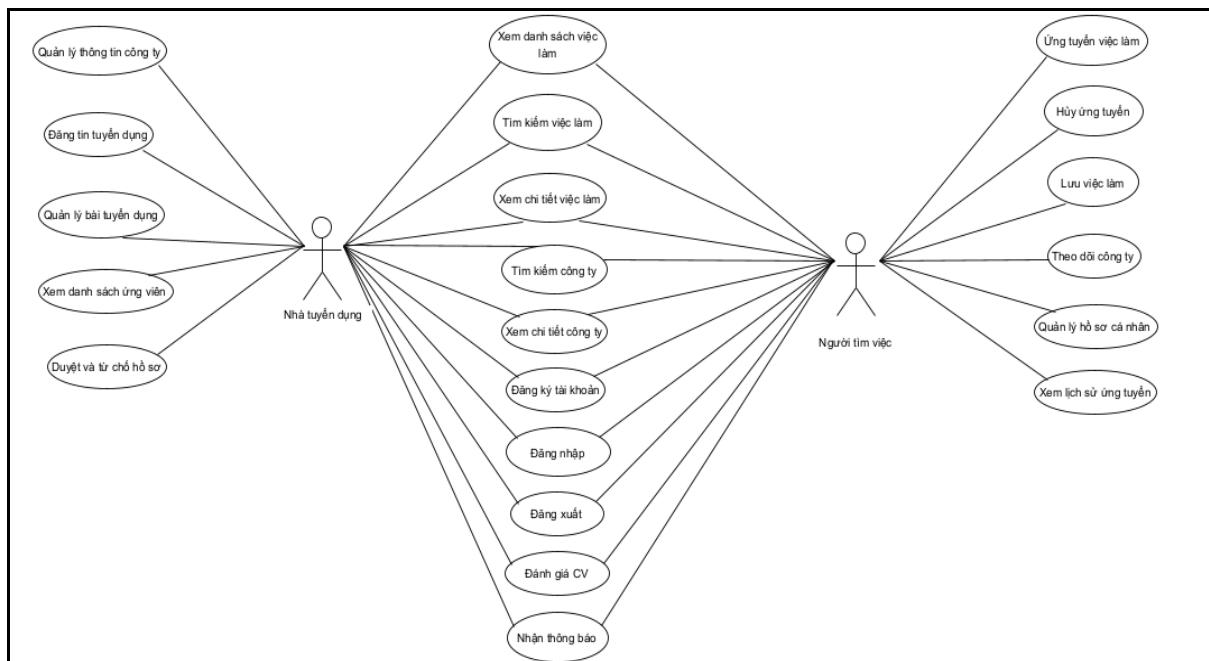
Hình 3.1 Sơ đồ phân cấp chức năng

Sơ đồ phân cấp chức năng của hệ thống website tìm kiếm và ứng tuyển việc làm thể hiện cấu trúc chức năng của hệ thống theo dạng phân cấp, từ chức năng tổng quát đến các chức năng chi tiết.

Ở cấp cao nhất, hệ thống được chia thành các nhóm chức năng chính gồm: quản lý người tìm việc, quản lý nhà tuyển dụng, quản lý công việc, ứng tuyển việc làm, lưu và theo dõi, và thông báo. Mỗi nhóm chức năng đảm nhiệm một tập hợp nghiệp vụ riêng, hỗ trợ toàn diện cho quá trình tìm kiếm, đăng tuyển và ứng tuyển việc làm.

Sơ đồ giúp làm rõ phạm vi chức năng của hệ thống, đồng thời là cơ sở cho việc thiết kế và triển khai các chức năng chi tiết trong các bước phát triển tiếp theo.

### 3.4 Sơ đồ Use-case



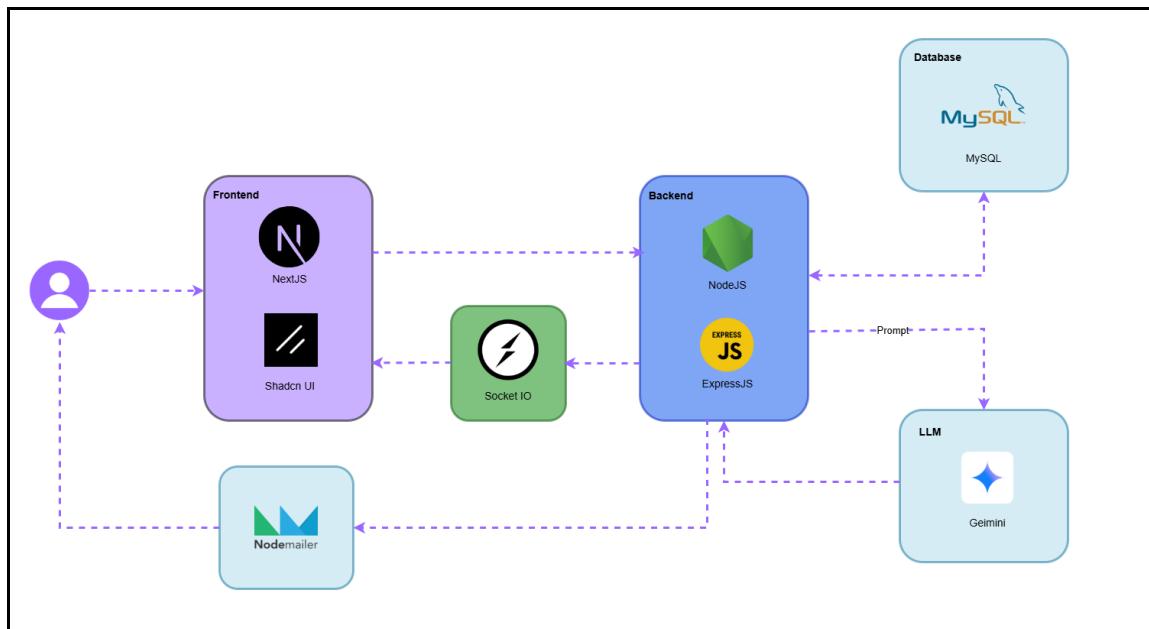
Hình 3.2: Sơ đồ Use-case

Người tìm việc có thể sử dụng đầy đủ các chức năng của hệ thống dành cho ứng viên, bao gồm: đăng ký tài khoản, đăng nhập, đăng xuất, xem và tìm kiếm việc làm, xem chi tiết việc làm và công ty, ứng tuyển việc làm, hủy ứng tuyển, lưu việc làm yêu thích, theo dõi công ty, quản lý hồ sơ cá nhân (thông tin cá nhân), xem lịch sử ứng tuyển, đánh giá CV và nhận thông báo từ hệ thống khi có trạng thái mới liên quan đến đơn ứng tuyển hoặc công ty theo dõi.

Nhà tuyển dụng có thể đăng ký và đăng nhập tài khoản doanh nghiệp, đăng xuất khỏi hệ thống, quản lý thông tin công ty, đăng tin tuyển dụng, quản lý và chỉnh sửa các bài tuyển dụng, xem danh sách ứng viên ứng tuyển cho từng công việc, duyệt hoặc từ chối hồ sơ ứng viên, đánh giá CV của ứng viên, tìm kiếm và xem danh sách việc làm,

công ty. Đồng thời nhận thông báo khi có ứng viên mới nộp hồ sơ hoặc các sự kiện liên quan đến hoạt động tuyển dụng.

### 3.5 Kiến trúc tổng thể của hệ thống



Hình 3.3: Kiến trúc hệ thống

Kiến trúc tổng thể của hệ thống website tìm kiếm và ứng tuyển việc làm, được xây dựng theo mô hình client–server, tách biệt rõ ràng giữa các thành phần giao diện, xử lý nghiệp vụ và lưu trữ dữ liệu gồm:

- **Frontend:** hệ thống sử dụng Next.js kết hợp với Shadcn UI để xây dựng giao diện người dùng hiện đại, thân thiện và có khả năng phản hồi nhanh. Người dùng tương tác với hệ thống thông qua trình duyệt, các yêu cầu được gửi đến backend thông qua giao thức HTTP/HTTPS và kết nối thời gian thực.

- **Backend:** Node.js cùng với Express.js đóng vai trò xử lý logic nghiệp vụ và cung cấp các API theo kiến trúc RESTful. Backend tiếp nhận yêu cầu từ frontend, xử lý xác thực, phân quyền, quản lý công việc, hồ sơ ứng tuyển và trả kết quả về cho người dùng.

- Hệ thống tích hợp Socket.IO nhằm hỗ trợ giao tiếp thời gian thực giữa frontend và backend, phục vụ các chức năng như nhận thông báo, cập nhật trạng thái ứng tuyển hoặc các sự kiện tức thời khác, giúp nâng cao trải nghiệm người dùng.

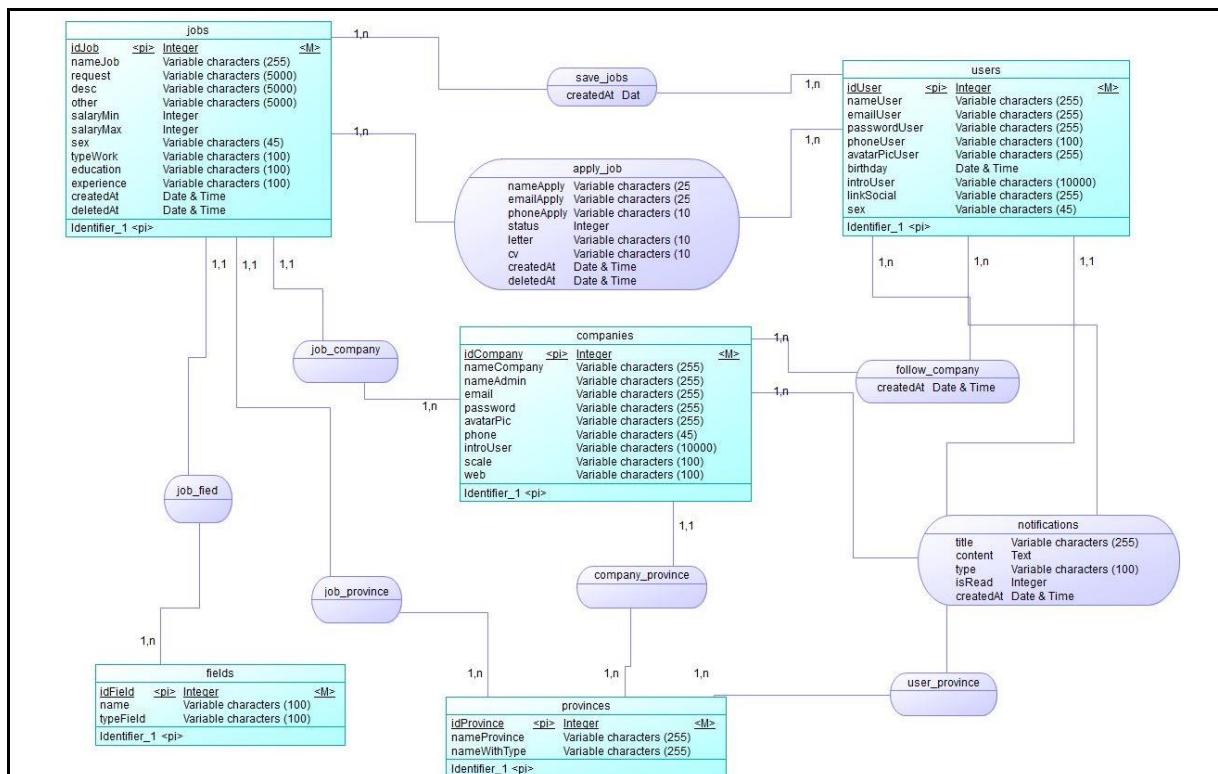
- Dữ liệu của hệ thống được lưu trữ tại MySQL, đóng vai trò là cơ sở dữ liệu trung tâm, đảm bảo khả năng lưu trữ có cấu trúc, truy vấn hiệu quả và tính toàn vẹn dữ liệu. Backend thực hiện các thao tác truy vấn và cập nhật dữ liệu thông qua các kết nối an toàn với cơ sở dữ liệu.

- Ngoài ra, hệ thống có khả năng tích hợp với mô hình ngôn ngữ lớn (LLM – Gemini) để thực hiện chức năng đánh giá CV người tìm việc với AI. Backend đóng vai trò trung gian gửi yêu cầu (prompt) và nhận kết quả từ LLM để phục vụ cho hệ thống.

Tổng thể, kiến trúc hệ thống được thiết kế theo hướng mô-đun, linh hoạt và dễ mở rộng, đảm bảo hiệu năng, khả năng bảo trì và phù hợp cho việc phát triển các chức năng nâng cao trong tương lai.

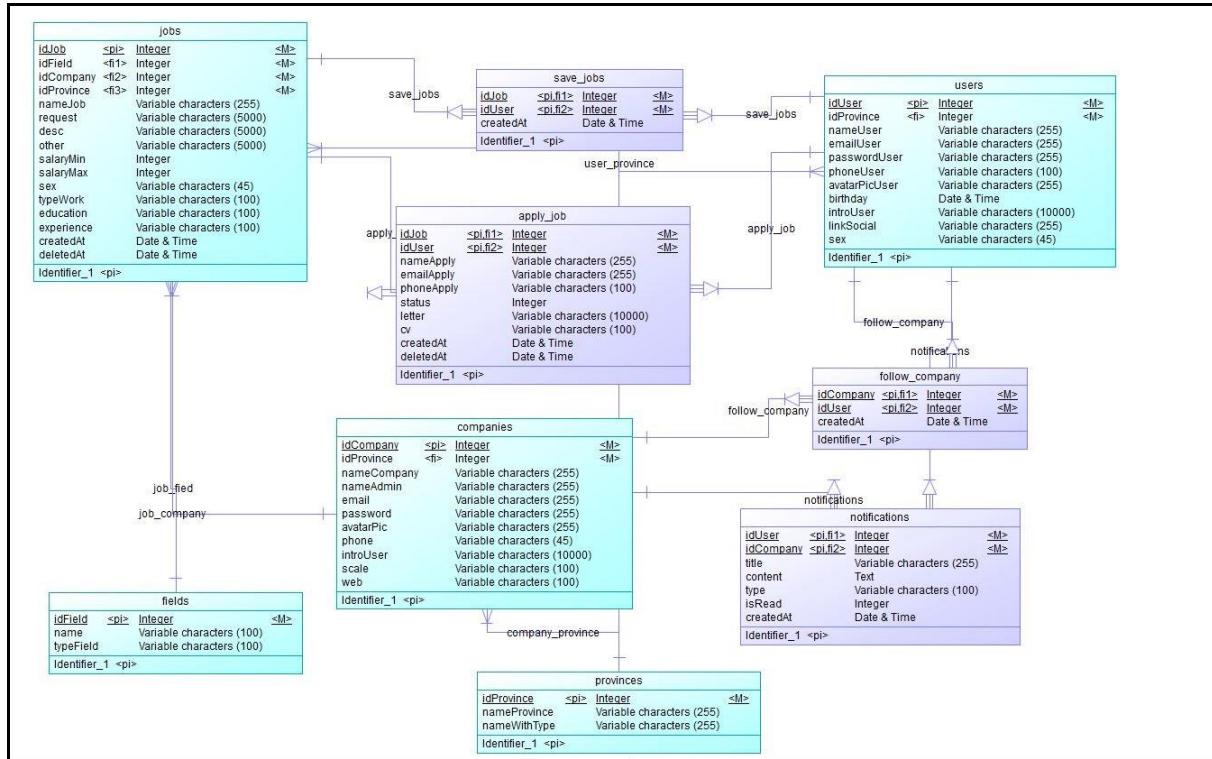
### 3.6 Thiết kế thành phần dữ liệu

#### 3.6.1 Mô hình dữ liệu mức quan niệm



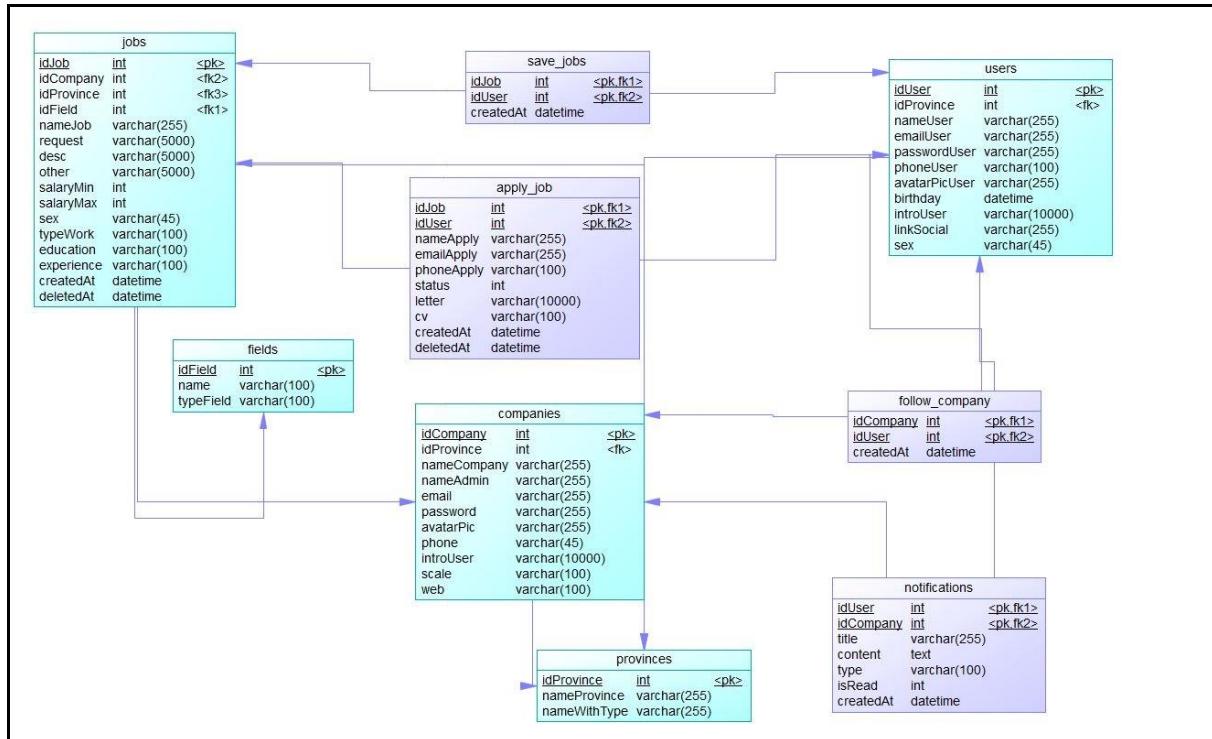
Hình 3.4: Mô hình dữ liệu mức quan niệm

### 3.6.2 Mô hình dữ liệu mức luận lý



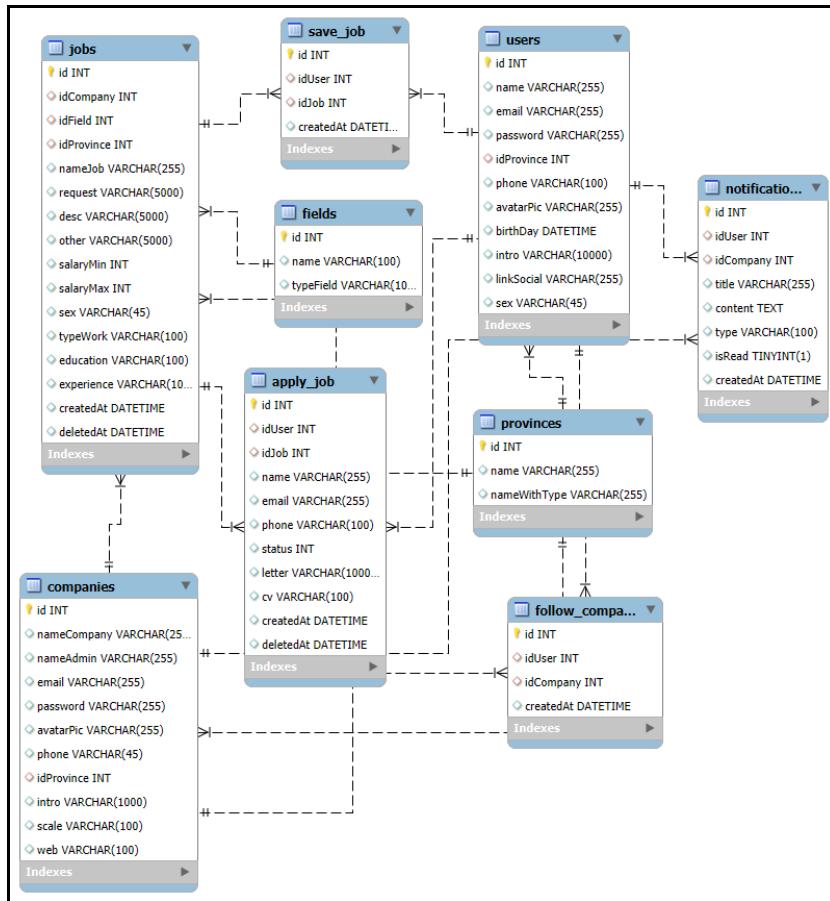
Hình 3.5: Mô hình dữ liệu mức luận lý

### 3.6.3 Mô hình dữ liệu mức vật lý



Hình 3.6: Mô hình dữ liệu mức vật lý

### 3.6.4 Lược đồ cơ sở dữ liệu



Hình 3.7: Lược đồ cơ sở dữ liệu

### 3.6.5 Thiết kế cơ sở dữ liệu

Bảng 3.1: Chi tiết thực thể “users”

STT	Thuộc tính	Diễn giải	Kiểu dữ liệu	Đặc điểm
1	id	Mã định danh của người tìm việc	int	<ul style="list-style-type: none"> <li>- Khóa chính (PK)</li> <li>-Not null</li> <li>- Tự động tăng (AUTO_INCREMENT)</li> <li>- Not null</li> </ul>
2	name	Họ tên người tìm việc	varchar(255)	- Not null
3	email	Email đăng nhập	varchar(255)	- Not null

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
4	password	Mật khẩu đã mã hóa	varchar(255)	- Not null
5	idProvince	Tỉnh/thành phố cư trú	int	- Khóa ngoại (FK)
6	phone	Số điện thoại	varchar(100)	
7	avatarPic	Ảnh đại diện	varchar(255)	
8	birthDay	Ngày sinh	datetime	
9	intro	Giới thiệu bản thân	varchar(1000 0)	
10	linkSocial	Liên kết mạng xã hội	varchar(255)	
11	sex	Giới tính	varchar(45)	

Bảng 3.2: Chi tiết thực thể “provinces”

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
1	id	Mã định danh tỉnh/thành phố	int	- Khóa chính (PK) - Not null - Tự động tăng (AUTO_INCREMENT)
2	name	Tên tỉnh/thành phố	varchar(255)	
3	nameWithType	Tên đầy đủ kèm loại hành chính	varchar(255)	

Bảng 3.3: Chi tiết thực thể “fields”

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
1	id	Mã ngành nghề	int	- Khóa chính (PK). - Not null - Tự động tăng

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
				(AUTO_INCREMENT)
2	name	Tên ngành nghề	varchar(100)	
3	typeField	Nhóm ngành nghề	varchar(100)	

Bảng 3.4: Chi tiết thực thể “companies”

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
1	id	Mã công ty	int	- Khóa chính (PK). - Not null - Tự động tăng (AUTO_INCREMENT)
2	nameCompany	Tên công ty	varchar(255)	
3	nameAdmin	Người đại diện	varchar(255)	
4	email	Email đăng nhập	varchar(255)	
5	password	Mật khẩu đã mã hóa	varchar(255)	
6	avatarPic	Logo công ty	varchar(255)	
7	phone	Số điện thoại	varchar(45)	
8	idProvince	Địa điểm hoạt động	int	- Khóa ngoại (FK)
9	intro	Giới thiệu công ty	varchar(1000)	
10	scale	Quy mô công ty	varchar(100)	
11	web	Website công ty	varchar(100)	

Bảng 3.5: Chi tiết thực thể “jobs”

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
1	id	Mã công việc	int	- Khóa chính (PK). - Not null - Tự động tăng (AUTO_INCREMENT)

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
2	idCompany	Công ty đăng tuyển	int	- Khóa ngoại (FK)
3	idField	Ngành nghề	int	- Khóa ngoại (FK)
4	idProvince	Địa điểm làm việc	int	- Khóa ngoại (FK)
5	nameJob	Tên công việc	varchar(255)	
6	request	Yêu cầu công việc	varchar(5000)	
7	desc	Mô tả công việc	varchar(5000)	
8	other	Quyền lợi khác	varchar(5000)	
9	salaryMin	Mức lương tối thiểu	int	
10	salaryMax	Mức lương tối đa	int	
11	sex	Yêu cầu giới tính	varchar(45)	
12	typeWork	Loại hình công việc	varchar(100)	
13	education	Trình độ học vấn	varchar(100)	
14	experience	Kinh nghiệm	varchar(100)	
15	createdAt	Ngày tạo	datetime	
16	deletedAt	Ngày xóa (soft delete)	datetime	

Bảng 3.6: Chi tiết thực thể “apply\_job”

<b>STT</b>	<b>Thuộc tính</b>	<b>Điễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
1	id	Mã đơn ứng tuyển	int	- Khóa chính (PK). - Not null - Tự động tăng

<b>STT</b>	<b>Thuộc tính</b>	<b>Diễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
				(AUTO_INCREMENT)
2	idUser	Người ứng tuyển	int	- Khóa ngoại (FK)
3	idJob	Công việc ứng tuyển	int	- Khóa ngoại (FK)
4	name	Tên ứng viên	varchar(255)	
5	email	Email ứng viên	varchar(255)	
6	phone	Số điện thoại	varchar(100)	
7	status	Trạng thái hồ sơ	int	
8	letter	Thư ứng tuyển	varchar(10000)	
9	cv	File CV	varchar(100)	
10	createdAt	Ngày nộp hồ sơ	datetime	
11	deletedAt	Ngày xóa	datetime	

Bảng 3.7: Chi tiết thực thể “save\_job”

<b>STT</b>	<b>Thuộc tính</b>	<b>Diễn giải</b>	<b>Kiểu dữ liệu</b>	<b>Đặc điểm</b>
1	id	Mã lưu việc	int	- Khóa chính (PK). - Not null - Tự động tăng (AUTO_INCREMENT)
2	idUser	Người lưu	int	- Khóa ngoại (FK)
3	idJob	Công việc được lưu	int	- Khóa ngoại (FK)
4	createdAt	Ngày lưu	datetime	

Bảng 3.8: Chi tiết thực thể “follow\_company”

STT	Thuộc tính	Điễn giải	Kiểu dữ liệu	Đặc điểm
1	id	Mã theo dõi	int	- Khóa chính (PK). - Not null - Tự động tăng (AUTO_INCREMENT)
2	idUser	Người theo dõi	int	- Khóa ngoại (FK)
3	idCompany	Công ty được theo dõi	int	- Khóa ngoại (FK)
4	createdAt	Ngày theo dõi	datetime	

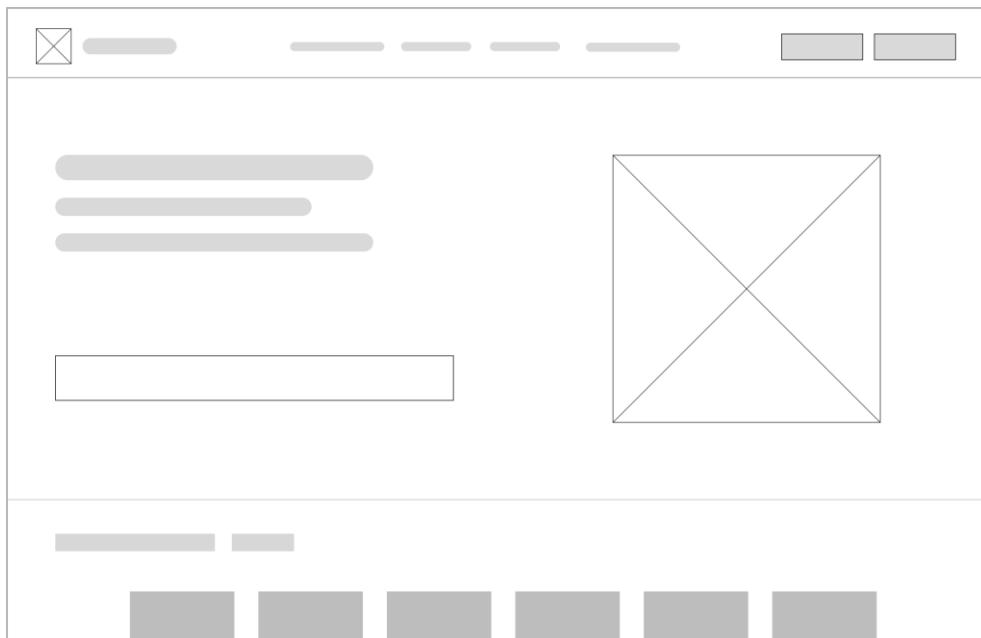
Bảng 3.9: Chi tiết thực thể “notifications”

STT	Thuộc tính	Điễn giải	Kiểu dữ liệu	Đặc điểm
1	id	Mã thông báo	int	- Khóa chính (PK). - Not null - Tự động tăng (AUTO_INCREMENT)
2	idUser	Người nhận	int	- Khóa ngoại (FK)
3	idCompany	Công ty gửi (nếu có)	int	- Khóa ngoại (FK)
4	title	Tiêu đề thông báo	varchar(255)	
5	content	Nội dung thông báo	text	
6	type	Loại thông báo	varchar(100)	
7	isRead	Trạng thái đã đọc	boolean	Mặc định = 0
8	createdAt	Thời điểm tạo	datetime	

### 3.7 Thiết kế giao diện

#### 3.7.1 Giao diện trang chủ

Giao diện trang chủ bao gồm thanh điều hướng chính, khu vực tìm kiếm nhanh, danh mục ngành nghề phổ biến, danh sách công ty tiêu biểu và các việc làm mới nhất. Cách sắp xếp này giúp người dùng dễ dàng tiếp cận thông tin, nhanh chóng tìm kiếm và ứng tuyển các vị trí phù hợp.



Hình 3.8: Thiết kế giao diện trang chủ

#### 3.7.2 Giao diện trang tìm việc nhanh

Giao diện trang ngành nghề và địa điểm cho phép người dùng tìm kiếm việc làm nhanh chóng thông qua việc lựa chọn theo ngành nghề hoặc tỉnh/thành phố.



Hình 3.9: Thiết kế giao diện trang tìm việc nhanh

### 3.7.3 Giao diện trang tìm việc

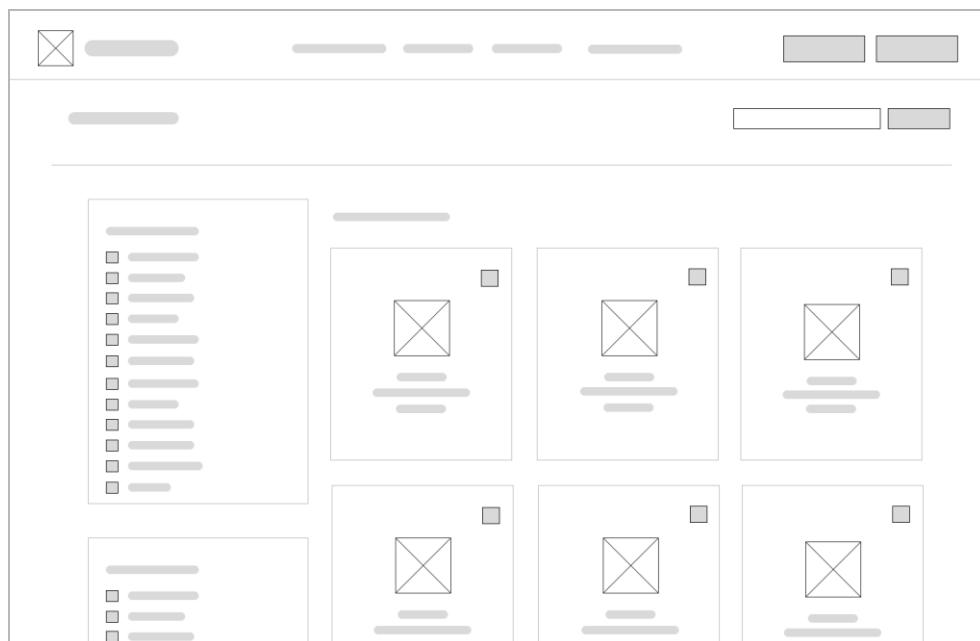
Giao diện trang tìm kiếm việc làm cho phép người dùng tra cứu và lọc các vị trí tuyển dụng theo nhiều tiêu chí như từ khóa, tỉnh/thành phố, ngành nghề, mức lương, loại công việc, kinh nghiệm và học vấn. Kết quả tìm kiếm được hiển thị dưới dạng danh sách các thẻ công việc, cung cấp thông tin cơ bản và nút ứng tuyển nhanh.



Hình 3.10: Thiết kế giao diện trang tìm việc

### 3.7.4 Giao diện trang tìm kiếm công ty

Giao diện trang công ty hiển thị danh sách các nhà tuyển dụng nổi bật trên hệ thống, người dùng có thể tìm kiếm và lọc công ty theo địa điểm và quy mô nhân sự. Mỗi công ty được trình bày dưới dạng thẻ thông tin gồm logo, tên công ty, vị trí hoạt động và số lượng việc làm đang tuyển.



Hình 3.11: Thiết kế giao diện trang tìm kiếm công ty

### 3.7.5 Giao diện trang chi tiết công việc

Giao diện trang chi tiết việc làm hiển thị đầy đủ thông tin của một vị trí tuyển dụng, bao gồm tên công việc, nhà tuyển dụng, địa điểm, mức lương, hình thức làm việc, kinh nghiệm và học vấn yêu cầu.



Hình 3.12: Thiết kế giao diện trang chi tiết công việc

### 3.7.6 Giao diện trang chi tiết đơn ứng tuyển

Giao diện trang chi tiết đơn ứng tuyển hiển thị đầy đủ thông tin của một hồ sơ ứng tuyển, bao gồm thông tin ứng viên, vị trí ứng tuyển, thời gian nộp hồ sơ và trạng thái xử lý. Ngoài ra, hệ thống cung cấp các chức năng hỗ trợ nhà tuyển dụng như xem thư xin việc, tải CV và liên hệ trực tiếp với ứng viên.



Hình 3.13: Thiết kế giao diện trang chi tiết đơn ứng tuyển

### 3.7.7 Giao diện trang quản lý đơn ứng tuyển

Giao diện trang danh sách đơn xin việc đã nhận cho phép nhà tuyển dụng quản lý các hồ sơ ứng tuyển thông qua bảng danh sách trực quan. Hệ thống hỗ trợ tìm kiếm, lọc theo công việc và trạng thái, đồng thời cho phép sắp xếp các đơn ứng tuyển theo thời gian.



Hình 3.14: Thiết kế giao diện trang quản lý đơn ứng tuyển

### 3.7.8 Giao diện trang đăng bài tuyển dụng

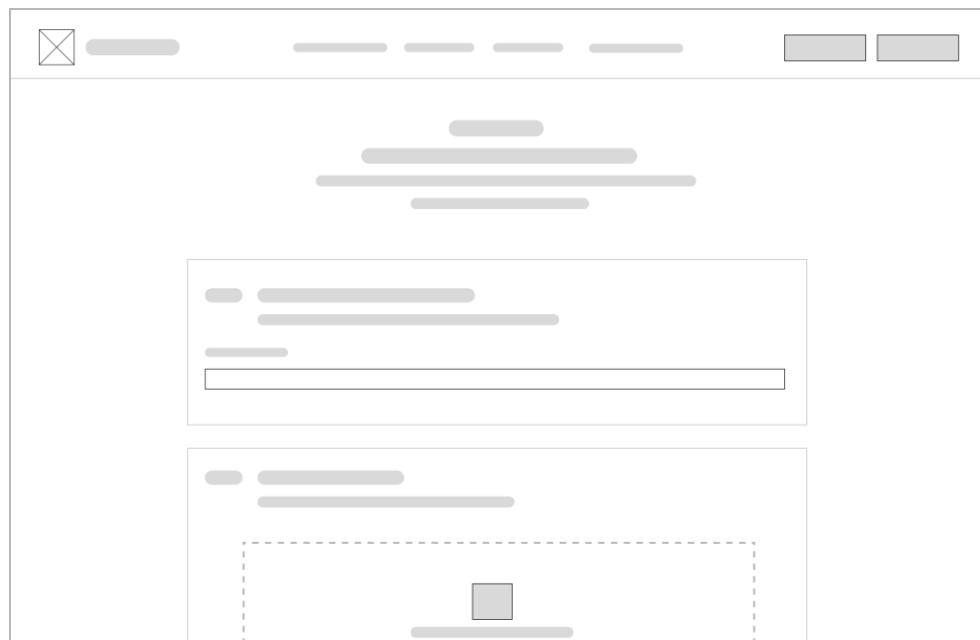
Giao diện trang đăng bài tuyển dụng cho phép nhà tuyển dụng tạo và quản lý thông tin tuyển dụng thông qua biểu mẫu nhập liệu trực quan. Trang cung cấp các trường thông tin cần thiết như chức danh, ngành nghề, địa điểm làm việc, mức lương, hình thức làm việc, yêu cầu công việc và mô tả chi tiết.



Hình 3.15: Thiết kế giao diện trang đăng bài tuyển dụng

### 3.7.9 Giao diện trang đánh giá CV với AI

Giao diện trang đánh giá CV với AI được thiết kế nhằm hỗ trợ cả người tìm việc và nhà tuyển dụng trong việc phân tích và đánh giá chất lượng hồ sơ xin việc một cách tự động. Người dùng có thể chọn vị trí công việc mong muốn và tải lên tệp CV, hệ thống sẽ sử dụng trí tuệ nhân tạo để phân tích nội dung CV dựa trên các tiêu chí như kỹ năng, kinh nghiệm, học vấn và mức độ phù hợp với vị trí ứng tuyển.



Hình 3.16: Thiết kế giao diện trang đánh giá CV với AI

### 3.8 Triển khai

Triển khai ứng dụng với Docker là một cách hiệu quả để đảm bảo tính nhất quán và dễ dàng quản lý giữa các môi trường. Bằng cách sử dụng Dockerfile và Docker Compose, chúng ta có thể đóng gói cả backend (Node.js, Express.js) và frontend (Next.js) vào các container riêng biệt, giúp việc triển khai trở nên đơn giản và đáng tin cậy.

Câu hình minh họa dockerfile cho backend:

```
FROM node:20-alpine

# Cài dependency hệ thống cần thiết (build + runtime)
RUN apk add --no-cache \
    python3 make g++ \
    cairo-dev jpeg-dev pango-dev \
    giflib-dev pixman-dev freetype-dev \
    tesseract-ocr \
    tesseract-ocr-data-eng \
    tesseract-ocr-data-vie \
    dumb-init curl

WORKDIR /app

# Copy package và cài dependency
COPY package*.json ./
RUN npm ci && npm cache clean --force

# Copy source và build
COPY .
RUN npm run build

# Tạo user non-root
RUN addgroup -S nodejs && adduser -S expressjs -G nodejs

# Thư mục upload
RUN mkdir -p uploads/{avatars,logos,cvs,temp} \
    && chown -R expressjs:nodejs uploads \
    && chmod -R 755 uploads

USER expressjs

EXPOSE 5000

ENV NODE_ENV=production
ENV PORT=5000

HEALTHCHECK --interval=30s --timeout=10s \
    CMD curl -f http://localhost:5000/api/health || exit 1

ENTRYPOINT ["dumb-init","--"]
```

```
CMD ["node", "dist/server.js"]
```

Câu hình minh họa dockerfile cho frontend:

```
FROM node:20-alpine

RUN apk add --no-cache libc6-compat curl
WORKDIR /app

# Cài dependency
COPY package*.json .
RUN npm ci && npm cache clean --force

# Copy source + build
COPY . .

ARG NEXT_PUBLIC_API_URL
ARG NEXT_PUBLIC_BACKEND_URL

ENV NEXT_PUBLIC_API_URL=$NEXT_PUBLIC_API_URL
ENV NEXT_PUBLIC_BACKEND_URL=$NEXT_PUBLIC_BACKEND_URL
ENV NODE_ENV=production
ENV PORT=3000

RUN npm run build

# Non-root user
RUN addgroup -S nodejs && adduser -S nextjs -G nodejs
USER nextjs

EXPOSE 3000

HEALTHCHECK --interval=30s --timeout=3s \
CMD curl -f http://localhost:3000/api/health || exit 1

CMD ["node", "server.js"]
```

Câu hình minh họa Docker Compose:

```
version: "3.8"

services:
  database:
    image: mysql:8.0
    container_name: jobify-db
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD:-root}
      MYSQL_DATABASE: ${DB_NAME:-jobify_db}
      MYSQL_USER: ${DB_USER:-jobify_user}
      MYSQL_PASSWORD: ${DB_PASSWORD:-jobify_password}
    ports:
```

```
- "3306:3306"
volumes:
  - db_data:/var/lib/mysql
  - ./database:/docker-entrypoint-initdb.d:ro
networks: [jobify-net]

server:
  build: ./server
  container_name: jobify-server
  restart: unless-stopped
  environment:
    NODE_ENV: production
    PORT: 5000
    DB_HOST: database
    DB_PORT: 3306
    DB_NAME: ${DB_NAME:-jobify_db}
    DB_USER: ${DB_USER:-jobify_user}
    DB_PASSWORD: ${DB_PASSWORD:-jobify_password}
    JWT_SECRET: ${JWT_SECRET:-secret}
    FRONTEND_URL: ${FRONTEND_URL:-http://localhost:3000}
  ports:
    - "5000:5000"
  volumes:
    - server_uploads:/app/uploads
  depends_on: [database]
  networks: [jobify-net]

client:
  build:
    context: ./client
    args:
      NEXT_PUBLIC_API_URL: ${NEXT_PUBLIC_API_URL:-http://localhost:5000/api}
      NEXT_PUBLIC_BACKEND_URL: ${NEXT_PUBLIC_BACKEND_URL:-http://localhost:5000}
    container_name: jobify-client
    restart: unless-stopped
    ports:
      - "3000:3000"
    depends_on: [server]
    networks: [jobify-net]

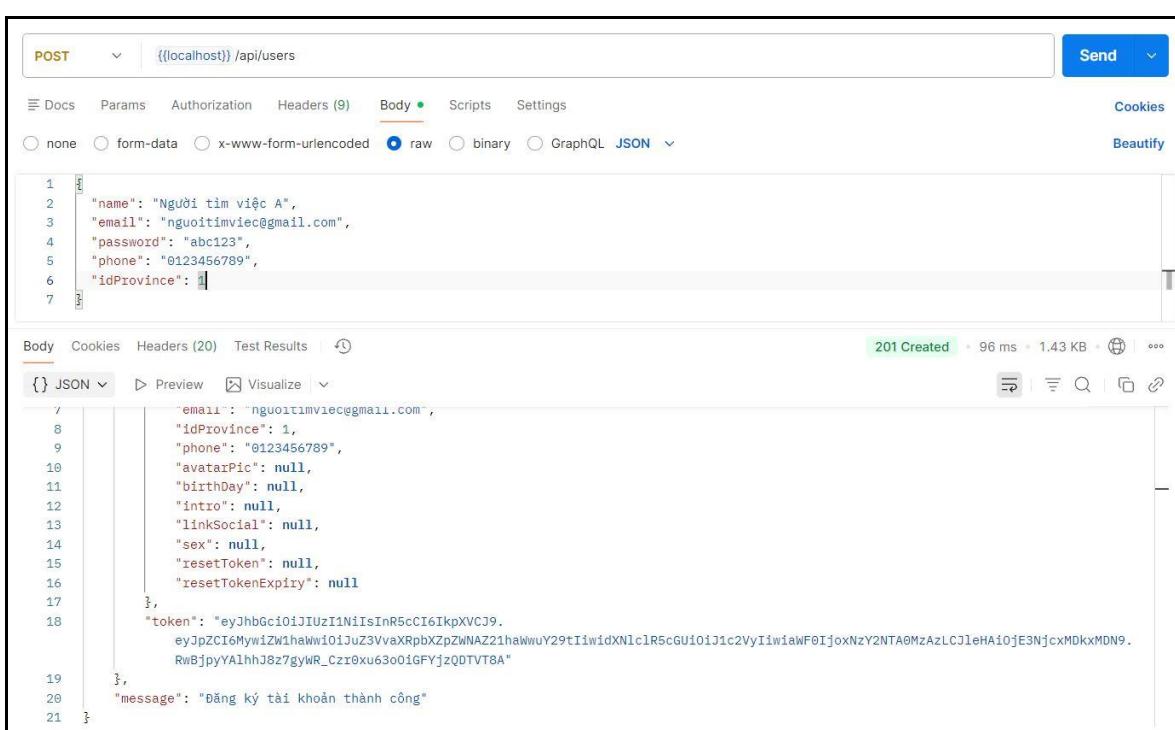
volumes:
  db_data:
  server_uploads:

networks:
  jobify-net:
    driver: bridge
```

## CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

### 4.1 Kết quả thử nghiệm API

Các API xác thực đóng vai trò quan trọng trong hệ thống tìm việc, có chức năng kiểm soát việc đăng ký, đăng nhập và xác thực danh tính người dùng trước khi truy cập vào các chức năng của hệ thống. Nhóm API này đảm bảo chỉ những người dùng hợp lệ mới có thể sử dụng các dịch vụ tương ứng với vai trò của mình, bao gồm người tìm việc và nhà tuyển dụng.

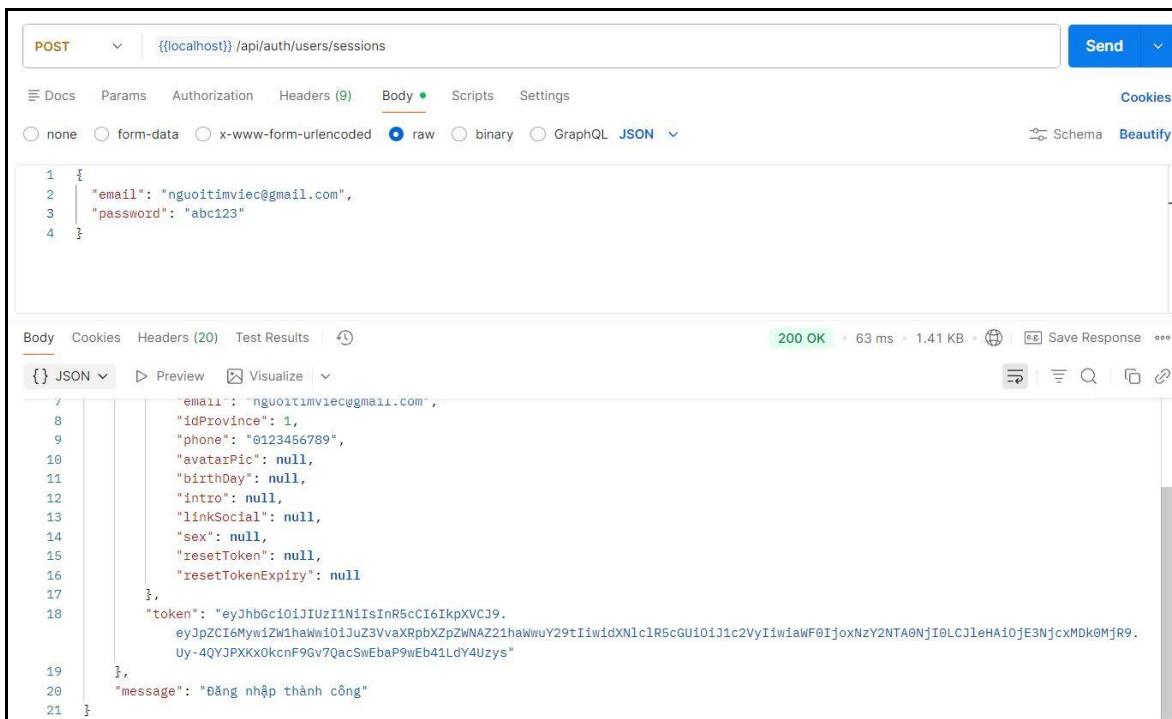


```

POST {{localhost}} /api/users
Body (raw) JSON
{
  "name": "Người tìm việc A",
  "email": "nguoitimviec@gmail.com",
  "password": "abc123",
  "phone": "0123456789",
  "idProvince": 1
}
201 Created
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Mjw1NjAwMjoiJuZ3VvaXRpZWNAZZ21hawwuY29tIiwidXNlclR5cGUiOiJ1c2VyIiwiaWF0IjoxNzY2NTA0MzA2LCJleHAiOjE3NjcxMDkxMDN9.RwBjpyYAlhhJ8z7gyWR_Czr0xu63o0IGFYjzQDVTvT8A",
  "message": "Đăng ký tài khoản thành công"
}
  
```

Hình 4.1: Thử nghiệm API tạo tài khoản người tìm việc

## Xây dựng website tìm kiếm và ứng tuyển việc làm



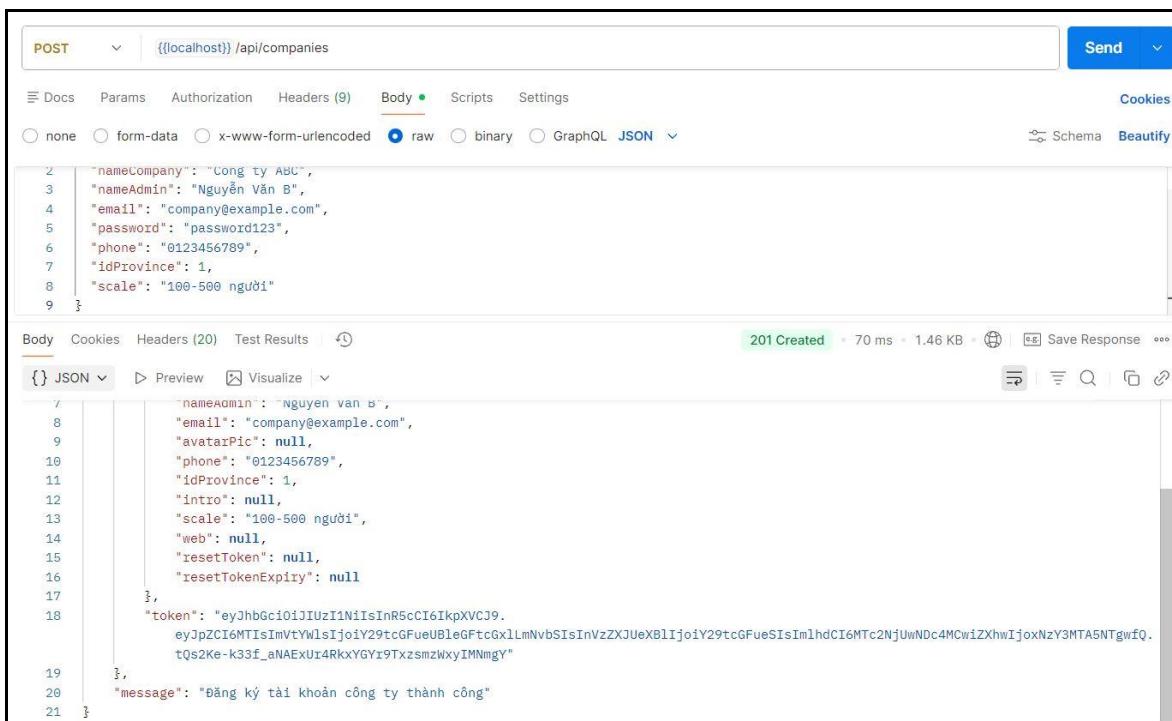
The screenshot shows a POST request to `http://localhost/api/auth/users/sessions`. The request body is a JSON object:

```
1 {
2     "email": "nguoitimviec@gmail.com",
3     "password": "abc123"
4 }
```

The response status is 200 OK, with a response time of 63 ms, size of 1.41 KB, and a message body containing a token and a success message.

```
200 OK • 63 ms • 1.41 KB • Save Response
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJpZCI6Myw1haww10iJu23VvaXRpbXZpZWAZ21haWWuY29tIiwidXNlclR5cGUi0iJ1c2VyiwiawF0IjoxNzY2NTA0NjI0LCJleHAi0jE3NjcxMDk0MjR9.
    ey-4QYJPXKx0kcnF9Gv7QacSwEbaP9mEb41LdY4Uzys",
    "message": "Đăng nhập thành công"
}
```

Hình 4.2: Thử nghiệm API đăng nhập tài khoản người tìm việc



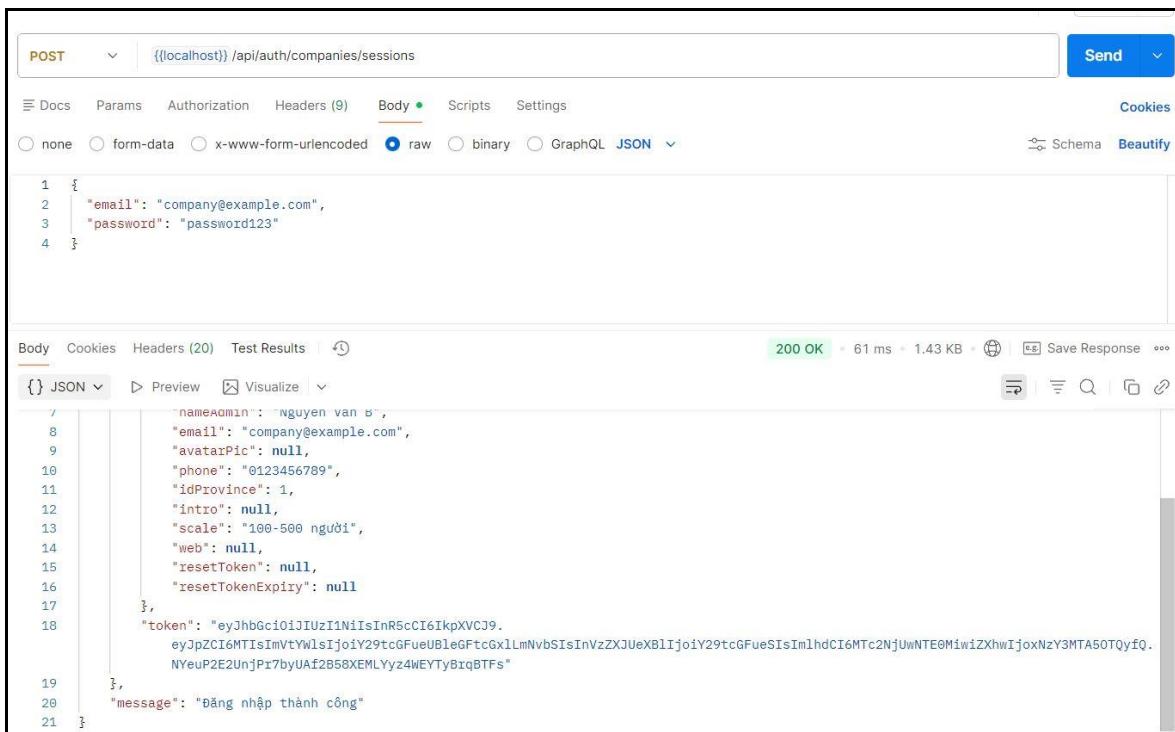
The screenshot shows a POST request to `http://localhost/api/companies`. The request body is a JSON object:

```
2 {
3     "nameCompany": "Công ty ABC",
4     "nameAdmin": "Nguyễn Văn B",
5     "email": "company@example.com",
6     "password": "password123",
7     "phone": "0123456789",
8     "idProvince": 1,
9     "scale": "100-500 người"
10 }
```

The response status is 201 Created, with a response time of 70 ms, size of 1.46 KB, and a response body containing a token and a success message.

```
201 Created • 70 ms • 1.46 KB • Save Response
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJpZCI6MTisImVtYwlSijoiY29tcGFueBleFtcGxlLmVbSISInVzZXJUeXBlijoiiY29tcGFueSISimhdCI6MTc2NjUwNDc4MCwiZXhwIjoxNzY3MTA5NTgwEQ.
    eyQs2Ke-k33f_aNAExUi4RkXYGYr9TxzsmzWxyIMNmgY",
    "message": "Đăng ký tài khoản công ty thành công"
}
```

Hình 4.3 Thử nghiệm API tạo tài khoản nhà tuyển dụng

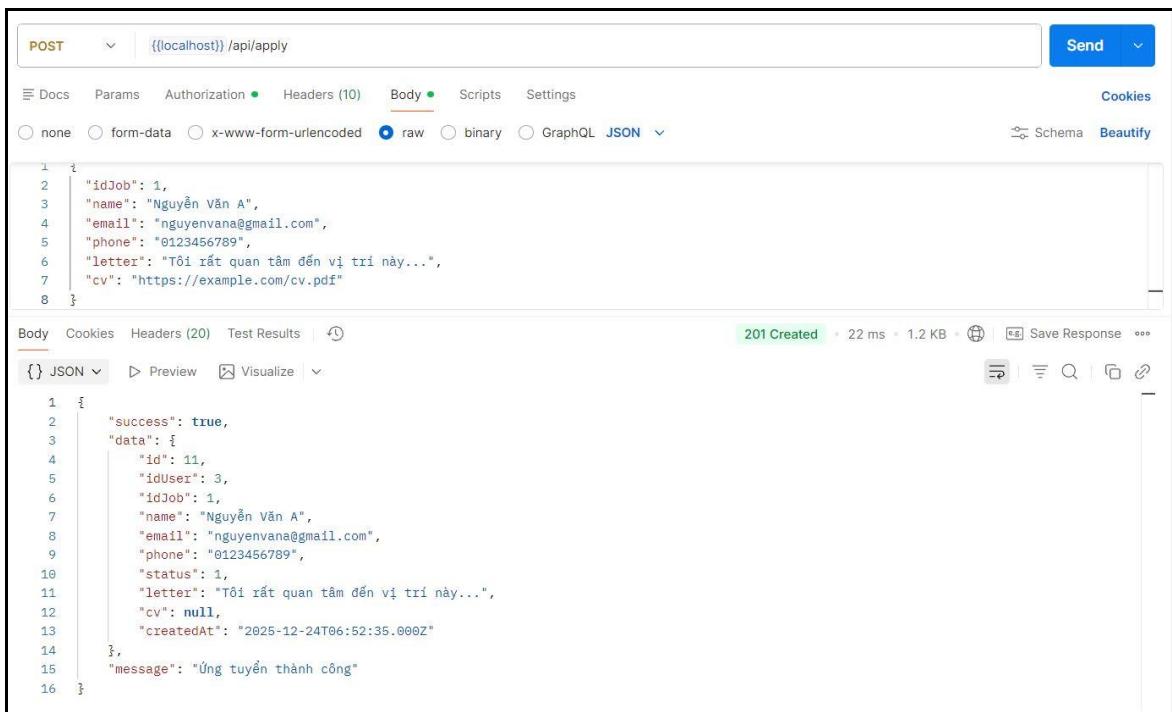


Hình 4.4 Thủ nghiệm API đăng nhập tài khoản nhà tuyển dụng

### 4.1.2 Các API ứng tuyển việc làm

Các API ứng tuyển việc làm được xây dựng nhằm hỗ trợ người tìm việc thực hiện và quản lý quá trình ứng tuyển trực tuyến trên hệ thống tìm việc. Nhóm API này cho phép người dùng gửi hồ sơ ứng tuyển vào các vị trí tuyển dụng, theo dõi danh sách các đơn đã nộp và thực hiện hủy ứng tuyển khi cần thiết.

## Xây dựng website tìm kiếm và ứng tuyển việc làm

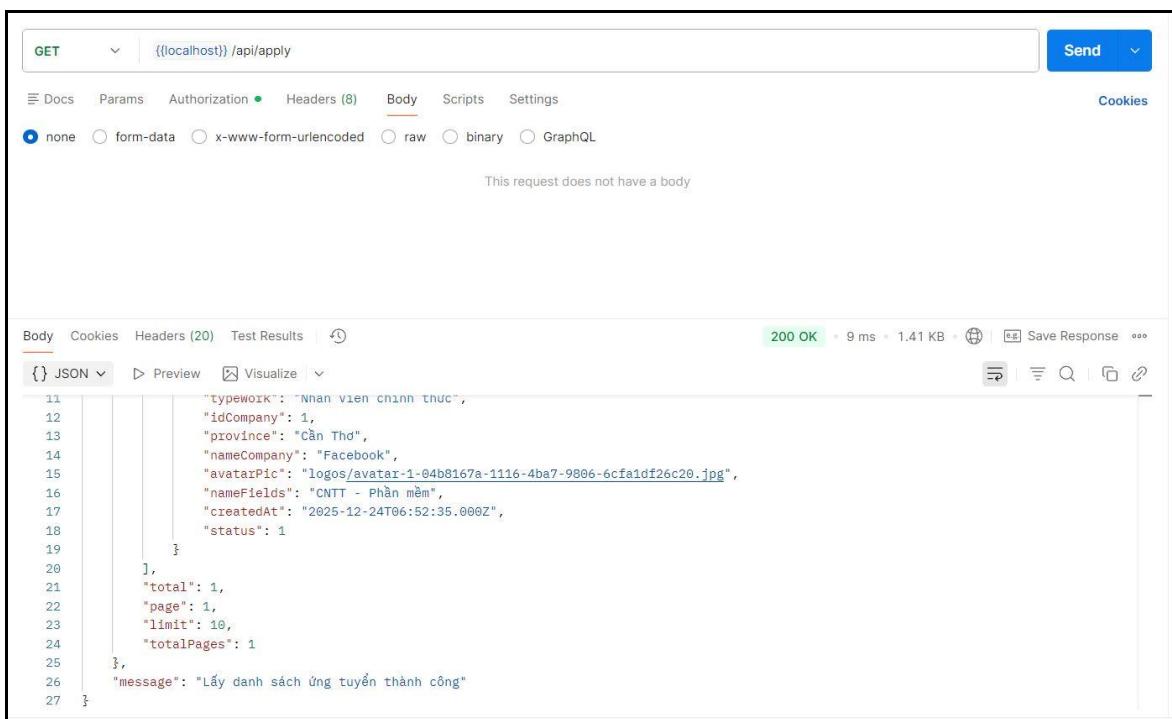


The screenshot shows a POST request to `{localhost} /api/apply`. The request body contains a JSON object representing a job application:

```
1  {
2   "idJob": 1,
3   "name": "Nguyễn Văn A",
4   "email": "nguyenvana@gmail.com",
5   "phone": "0123456789",
6   "letter": "Tôi rất quan tâm đến vị trí này...",
7   "cv": "https://example.com/cv.pdf"
8 }
```

The response status is 201 Created, with a response time of 22 ms, size of 1.2 KB, and a message "Ứng tuyển thành công".

Hình 4.5: Thử nghiệm API ứng tuyển công việc

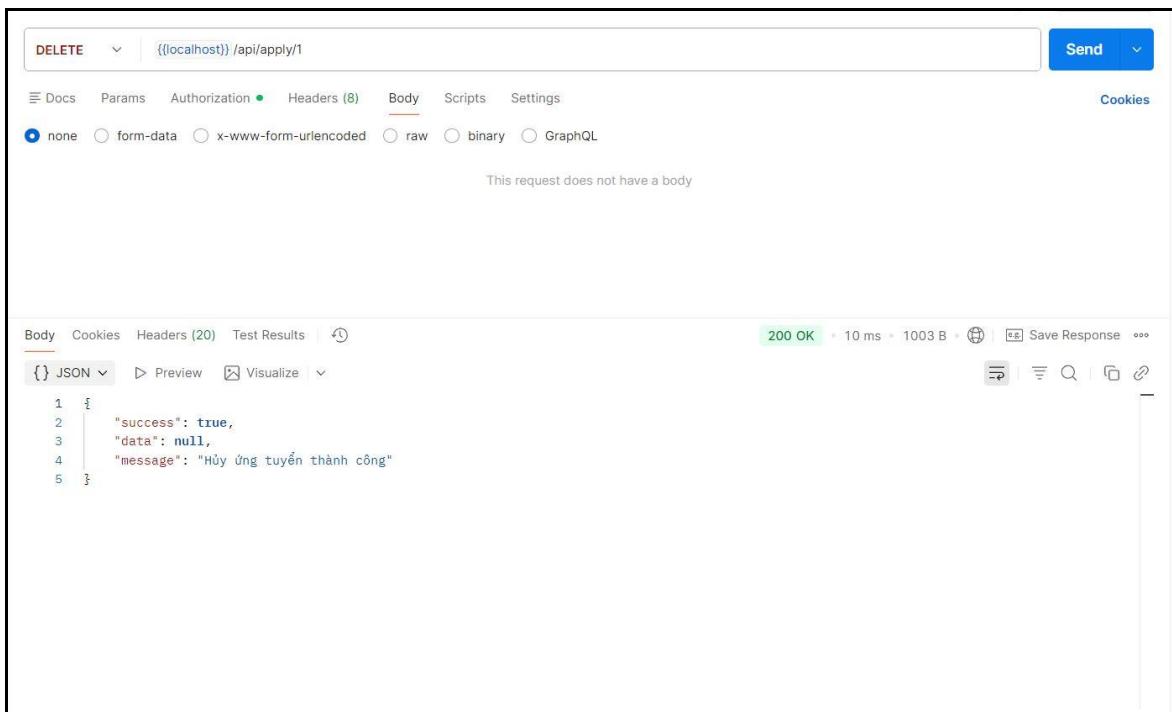


The screenshot shows a GET request to `{localhost} /api/apply`. The response status is 200 OK, with a response time of 9 ms, size of 1.41 KB, and a message "Lấy danh sách ứng tuyển thành công".

The response body is a JSON object containing a list of applications:

```
11   "typework": "Nhân viên chính thức",
12   "idCompany": 1,
13   "province": "Cần Thơ",
14   "nameCompany": "Facebook",
15   "avatarPic": "logos/avatar-1-04b8167a-1116-4ba7-9806-6cfaf1df26c20.jpg",
16   "nameFields": "CNTT - Phần mềm",
17   "createdAt": "2025-12-24T06:52:35.000Z",
18   "status": 1
19 }
20 ],
21 "total": 1,
22 "page": 1,
23 "limit": 10,
24 "totalPages": 1
25 },
26 "message": "Lấy danh sách ứng tuyển thành công"
27 }
```

Hình 4.6: Thử nghiệm API lấy danh sách đơn ứng tuyển của người tìm việc

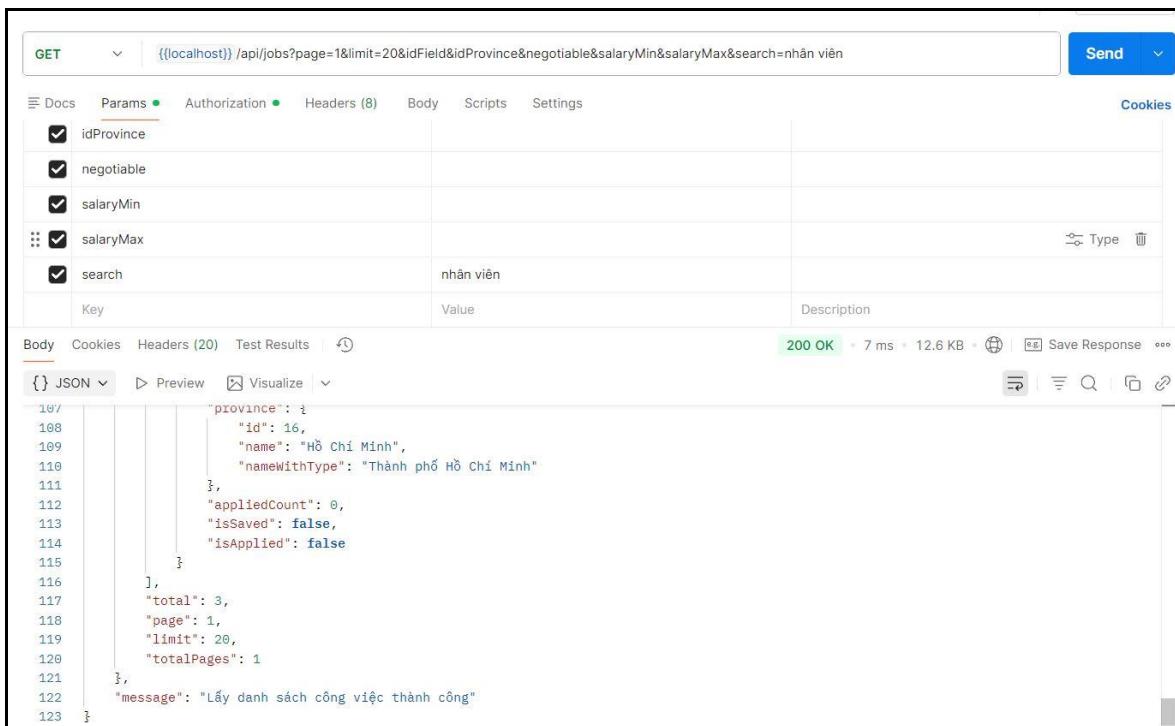


Hình 4.7: Thử nghiệm API hủy ứng tuyển công việc

### 4.1.3 Các API công việc

Các API công việc được xây dựng nhằm phục vụ chức năng quản lý và tra cứu thông tin tuyển dụng trên hệ thống tìm việc. Nhóm API này cho phép người tìm việc tìm kiếm và xem danh sách các công việc theo nhiều tiêu chí khác nhau, đồng thời hỗ trợ nhà tuyển dụng tạo mới và quản lý các tin tuyển dụng.

## Xây dựng website tìm kiếm và ứng tuyển việc làm



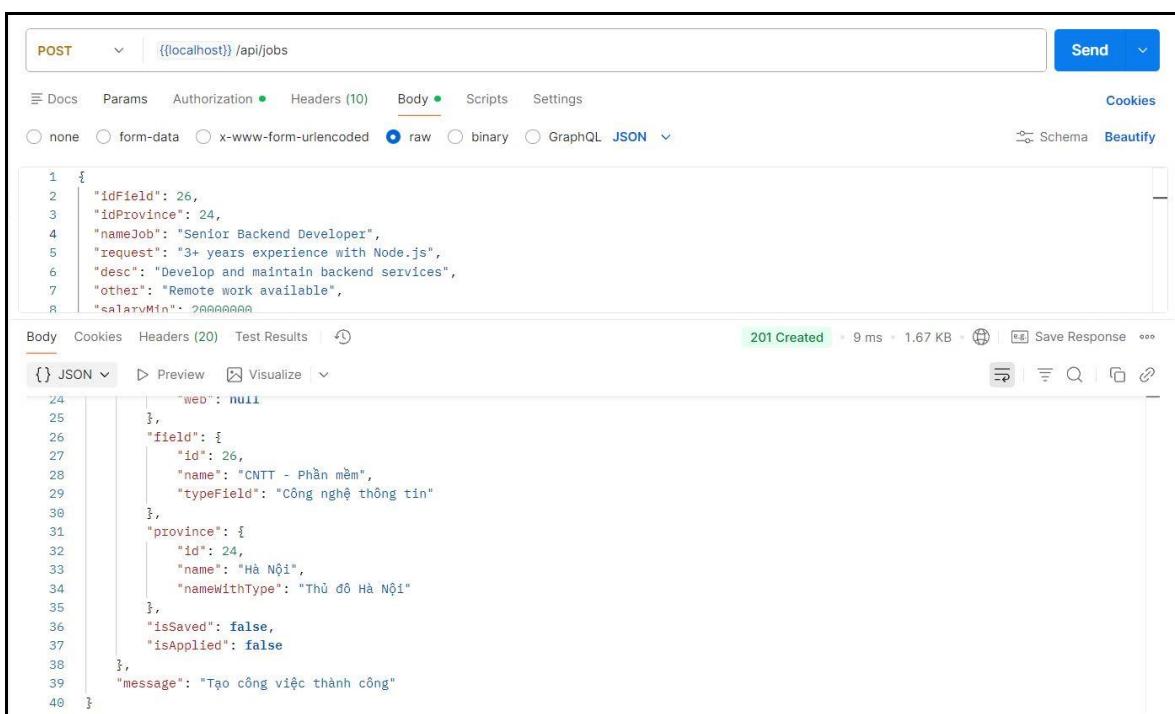
The screenshot shows a POST request to `{localhost}/api/jobs`. The request body is a JSON object:

```
1 {
2   "idField": 26,
3   "idProvince": 24,
4   "nameJob": "Senior Backend Developer",
5   "request": "3+ years experience with Node.js",
6   "desc": "Develop and maintain backend services",
7   "other": "Remote work available",
8   "salaryMin": 20000000
}
```

The response is a 201 Created status with JSON data:

```
1 {
2   "message": "Tạo công việc thành công"
}
```

Hình 4.8: Thử nghiệm API tìm kiếm công việc



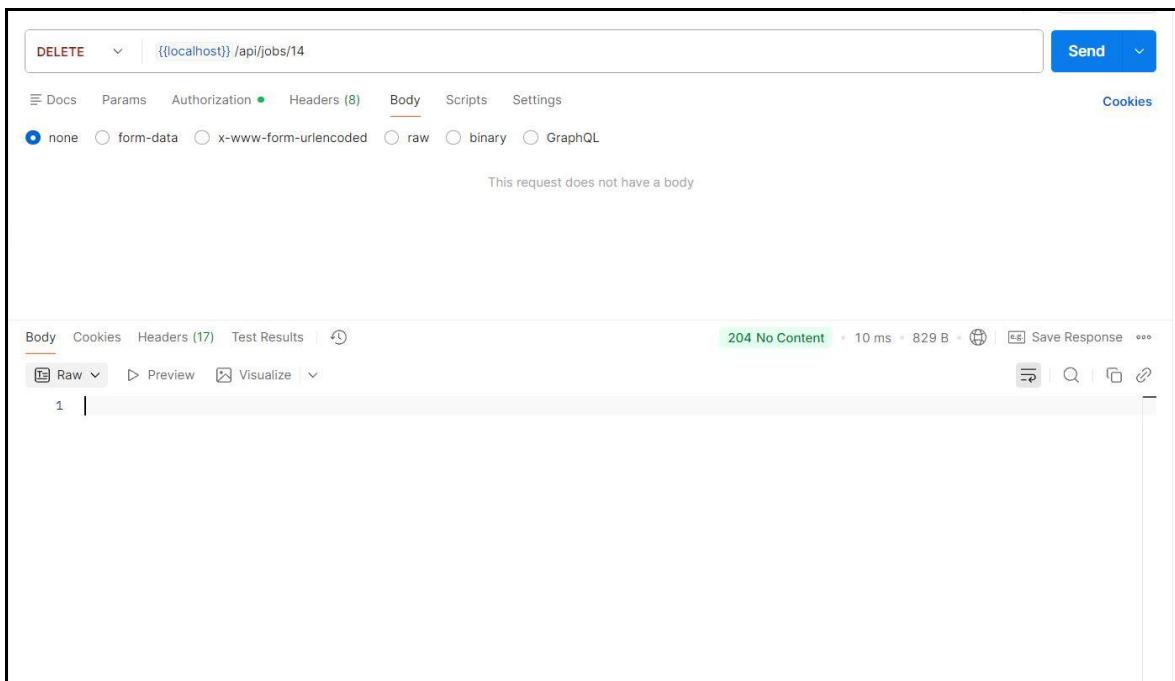
The screenshot shows a POST request to `{localhost}/api/jobs`. The request body is a JSON object:

```
1 {
2   "idField": 26,
3   "idProvince": 24,
4   "nameJob": "Senior Backend Developer",
5   "request": "3+ years experience with Node.js",
6   "desc": "Develop and maintain backend services",
7   "other": "Remote work available",
8   "salaryMin": 20000000
}
```

The response is a 201 Created status with JSON data:

```
1 {
2   "message": "Tạo công việc thành công"
}
```

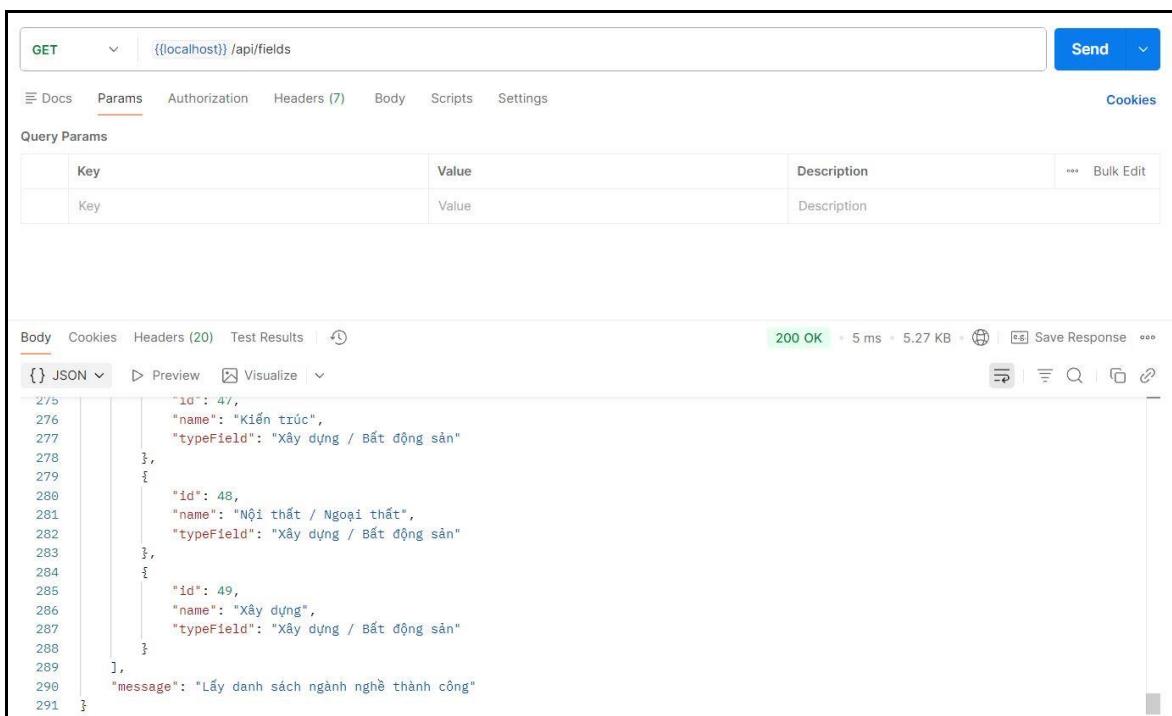
Hình 4.9: Thử nghiệm API tạo công việc cho nhà tuyển dụng



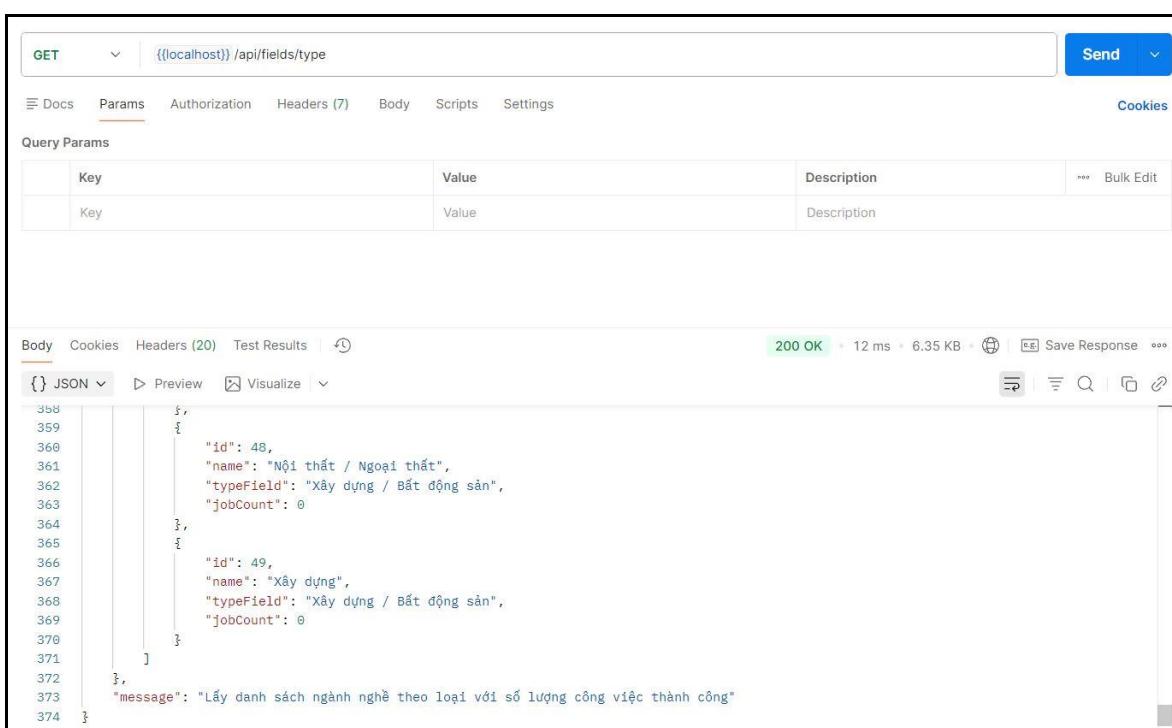
Hình 4.10: Thủ nghiệm API xóa công việc cho nhà tuyển dụng

#### 4.1.4 Các API ngành nghề

Các API ngành nghề được xây dựng nhằm phục vụ chức năng tra cứu và phân loại thông tin ngành nghề trong hệ thống tìm việc. Nhóm API này hỗ trợ việc lấy danh sách các ngành nghề hiện có, đồng thời cung cấp dữ liệu ngành nghề theo từng loại, giúp người tìm việc dễ dàng định hướng tìm kiếm và hỗ trợ nhà tuyển dụng trong quá trình đăng tin tuyển dụng.



Hình 4.11: Thủ nghiệm API lấy danh sách ngành nghề



Hình 4.12: Thủ nghiệm API lấy danh sách ngành nghề theo loại

#### 4.1.5 Các API nhà tuyển dụng

Các API nhà tuyển dụng được xây dựng nhằm phục vụ chức năng quản lý thông tin doanh nghiệp và hỗ trợ người tìm việc tra cứu nhà tuyển dụng trên hệ thống tìm việc. Nhóm API này cho phép lấy danh sách các công ty đang hoạt động,

truy xuất thông tin chi tiết của doanh nghiệp hiện tại và cập nhật thông tin nhà tuyển dụng.

```

{
    "data": [
        {
            "id": 149,
            "nameCompany": "Công ty ABC",
            "nameAdmin": "Nguyễn Văn B",
            "email": "company@example.com",
            "avatarPic": null,
            "phone": "0123456789",
            "idProvince": 1,
            "intro": null,
            "scale": "100-500 người",
            "web": null,
            "resetToken": null,
            "resetTokenExpIzy": null,
            "provinceName": "Tuyên Quang",
            "provinceFullName": "Tỉnh Tuyên Quang"
        }
    ],
    "total": 12,
    "page": 1,
    "limit": 10,
    "totalPages": 2
},
"message": "Lấy danh sách công ty thành công"
}

```

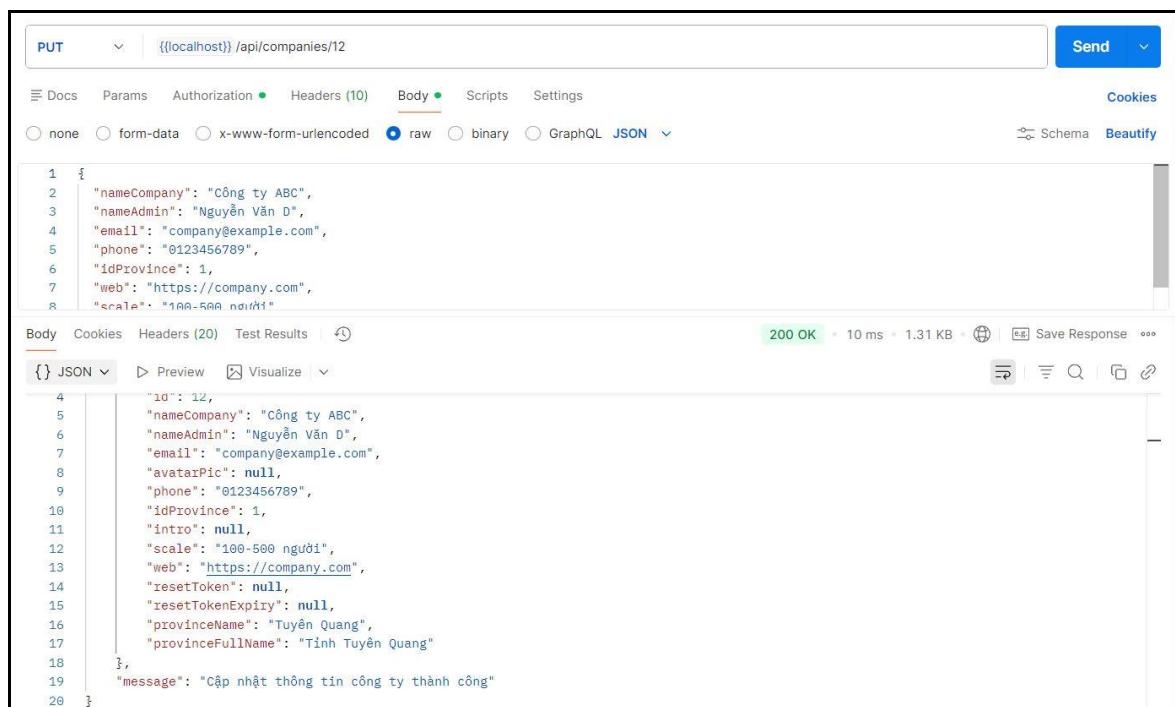
Hình 4.13: Thử nghiệm API lấy danh sách tất cả công ty

```

{
    "id": 12,
    "nameCompany": "Công ty ABC",
    "nameAdmin": "Nguyễn Văn B",
    "email": "company@example.com",
    "avatarPic": null,
    "phone": "0123456789",
    "idProvince": 1,
    "intro": null,
    "scale": "100-500 người",
    "web": null,
    "resetToken": null,
    "resetTokenExpIzy": null,
    "provinceName": "Tuyên Quang",
    "provinceFullName": "Tỉnh Tuyên Quang"
},
"message": "Lấy thông tin công ty hiện tại thành công"
}

```

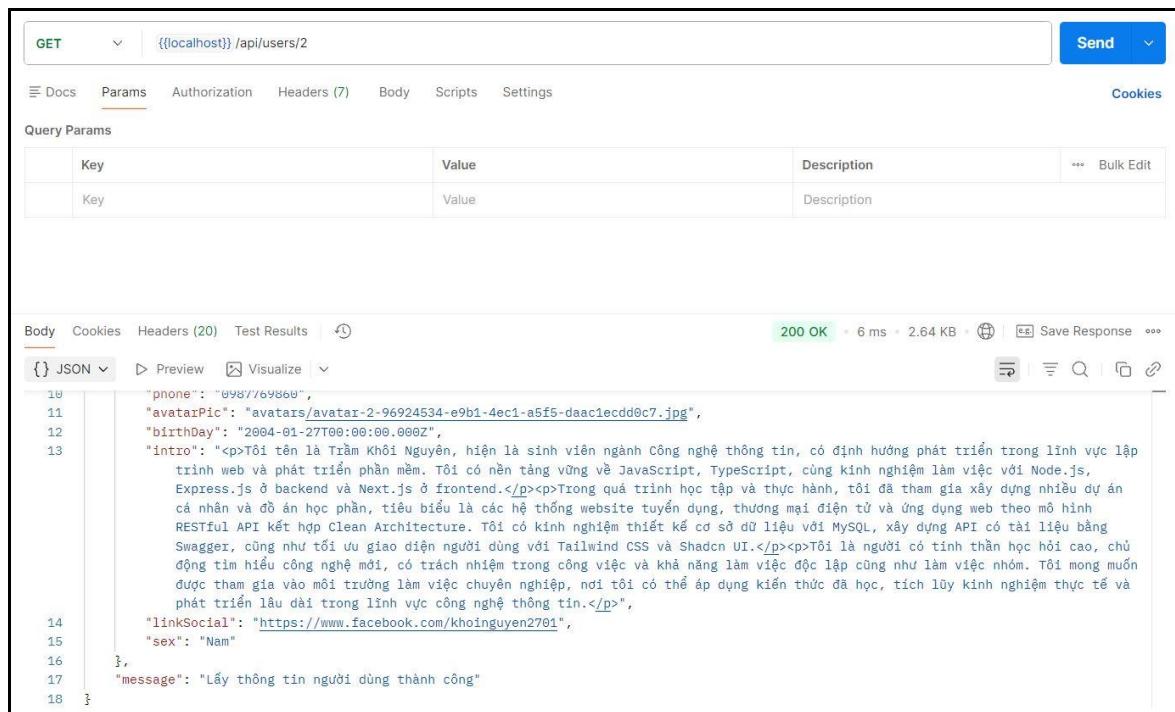
Hình 4.14: Thử nghiệm API lấy thông tin công ty hiện tại



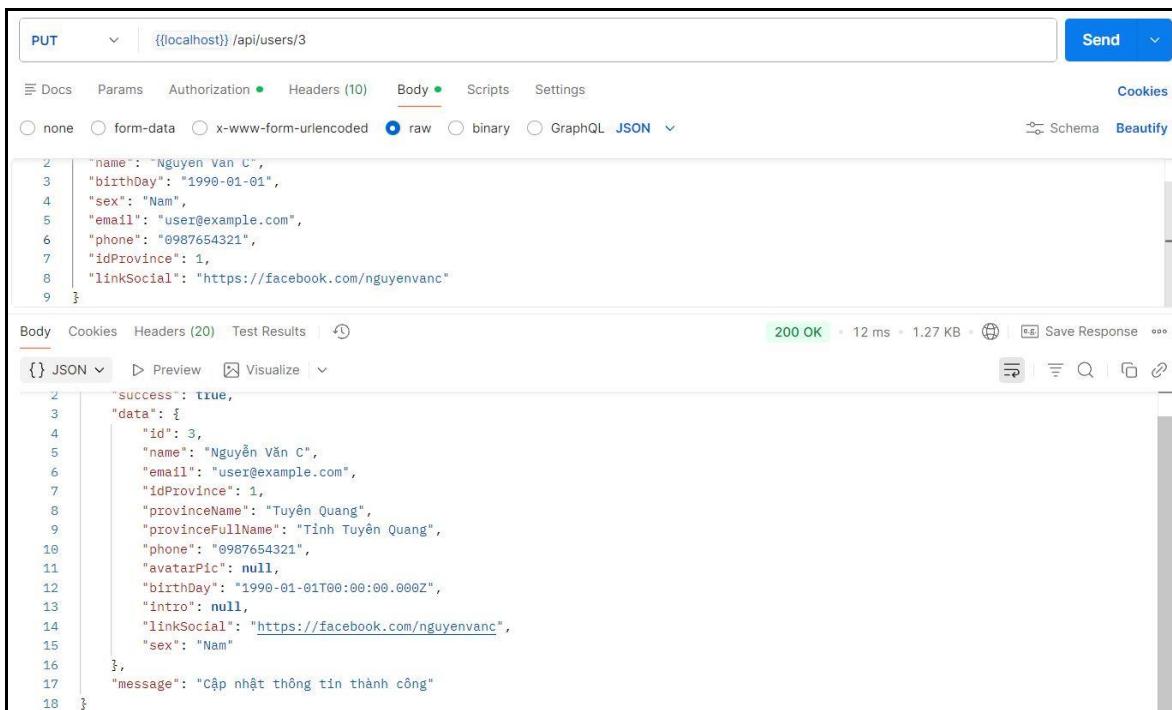
Hình 4.15: Thử nghiệm API cập nhật thông tin công ty

#### 4.1.6 Các API người tìm việc

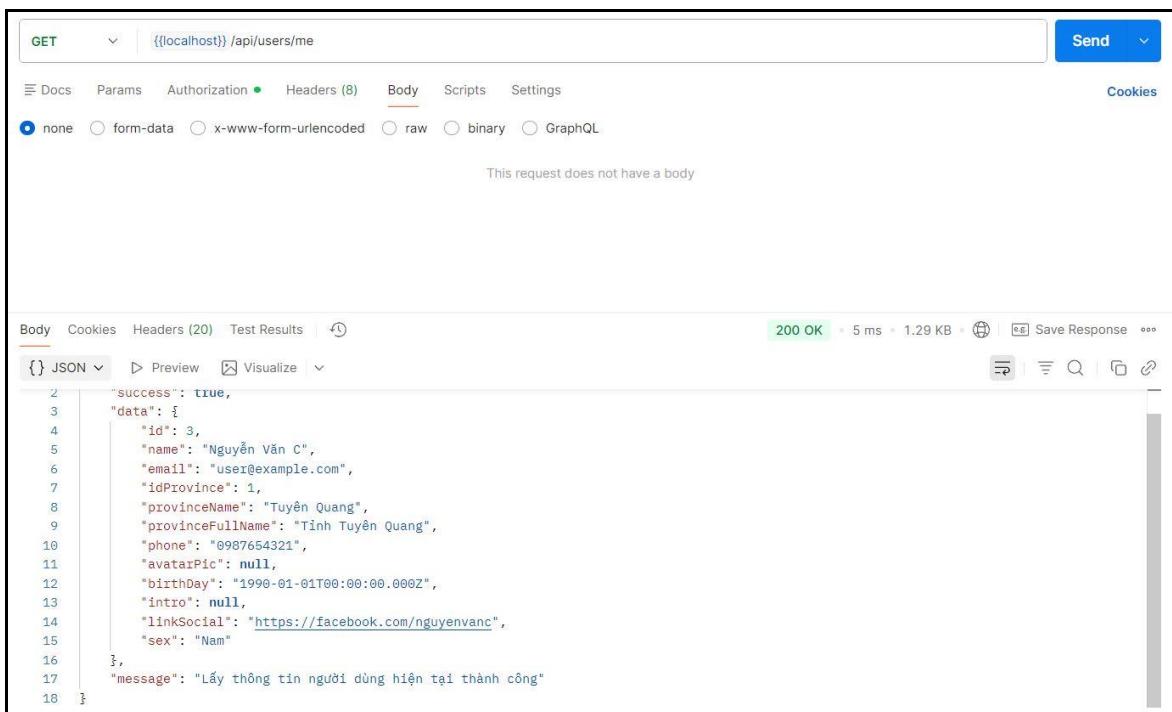
Các API người tìm việc được xây dựng nhằm phục vụ chức năng quản lý thông tin cá nhân của ứng viên trên hệ thống tìm việc. Nhóm API này cho phép truy xuất thông tin người tìm việc theo từng trường hợp, bao gồm lấy thông tin theo ID, lấy thông tin tài khoản hiện tại và cập nhật hồ sơ cá nhân.



Hình 4.16: Thử nghiệm API lấy thông tin người tìm việc theo ID



Hình 4.17: Thử nghiệm API cập nhật thông tin người tìm việc



Hình 4.18: Thử nghiệm API lấy thông tin người tìm việc hiện tại

#### 4.1.7 API đánh giá CV với AI

API đánh giá CV với AI được xây dựng nhằm hỗ trợ người tìm việc phân tích và đánh giá mức độ phù hợp của hồ sơ xin việc so với vị trí tuyển dụng mong muốn thông qua việc ứng dụng trí tuệ nhân tạo. API cho phép người dùng tải lên tệp

CV và lựa chọn công việc mục tiêu để hệ thống tiến hành xử lý và phân tích nội dung hồ sơ.

The screenshot shows a POST request to `http://{{localhost}}/api/cv-score`. The request body is set to `form-data` and contains two fields: `cvFile` (selected file: Trâm\_Khôi\_Nguyễn\_cv.pdf) and `jobId` (value: 1). The response status is `200 OK` with a response time of 22.11 s and a size of 3.73 KB. The response body is a JSON object:

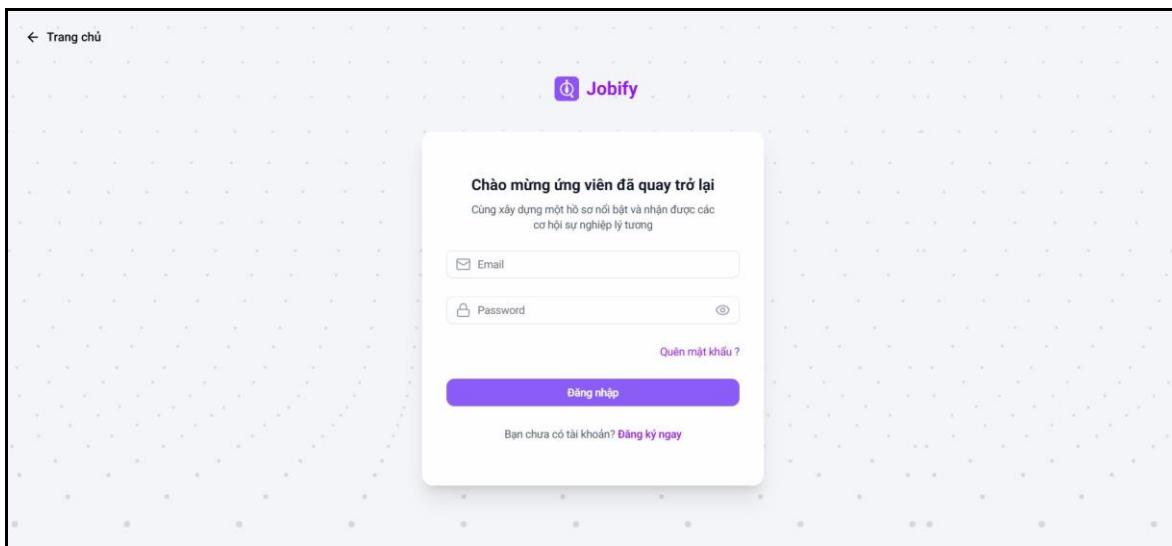
```
33     "Kinh nghiệm A/B testing và xây dựng pipeline dữ liệu cho phân tích",
34     "Kinh nghiệm phân tích hành vi người dùng trên các sản phẩm công nghệ"
35   ],
36   "jobMatch": {
37     "jobTitle": "Chuyên viên Phân tích Dữ liệu (Data Analyst)",
38     "companyName": "Facebook",
39     "requirements": [
40       "Kinh nghiệm thực tế của ứng viên chủ yếu là phát triển phần mềm (~10 tháng), không phù hợp với yêu cầu 2-5 năm kinh
41       nghiệm phân tích dữ liệu. Thông tin \"hơn 4 năm\" trong phần giới thiệu gây hiểu lầm nghiêm trọng.",
42       "Ứng viên đang theo học ngành CNTT, phù hợp về lĩnh vực nhưng chưa tốt nghiệp Đại học (dự kiến 2026), không đáp ứng yêu
43       cầu đã tốt nghiệp."
44     ]
45   },
46   "message": "Chấm điểm CV thành công"
47 }
```

Hình 4.19: Thử nghiệm API đánh giá hồ sơ xin việc với AI

## 4.2 Các giao diện của hệ thống

### 4.2.1 Giao diện đăng nhập người tìm việc

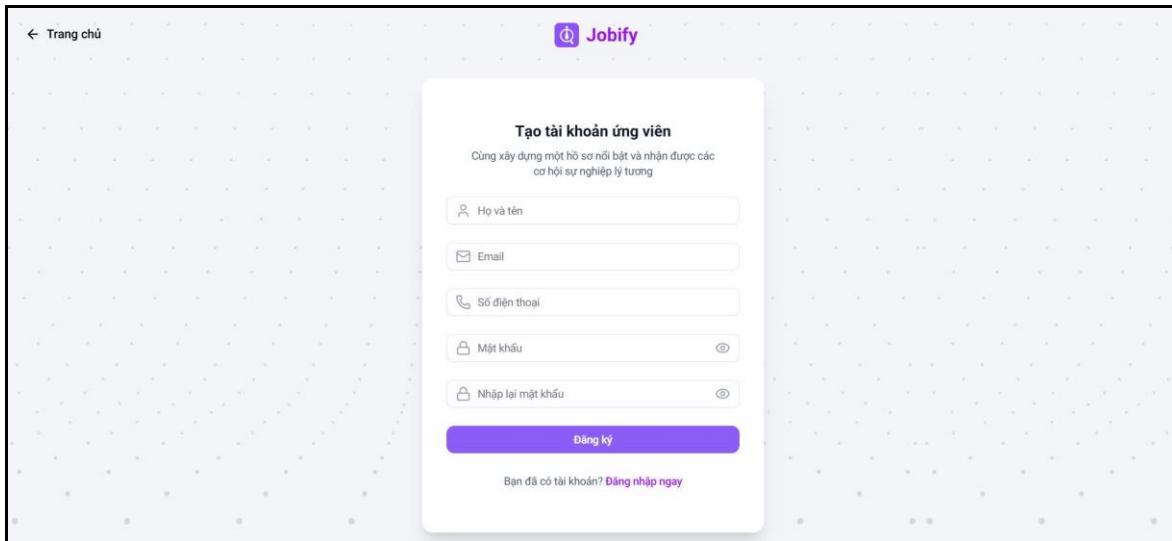
Giao diện đăng nhập người tìm việc là thành phần đầu tiên mà ứng viên cần thực hiện để có thể truy cập và sử dụng đầy đủ các chức năng của hệ thống tìm việc. Chức năng này nhằm xác thực danh tính người dùng, đảm bảo an toàn thông tin cá nhân và phân quyền truy cập đúng vai trò trong hệ thống.



Hình 4.20: Giao diện trang đăng nhập tài khoản người tìm việc

#### 4.2.2 Giao diện đăng ký người tìm việc

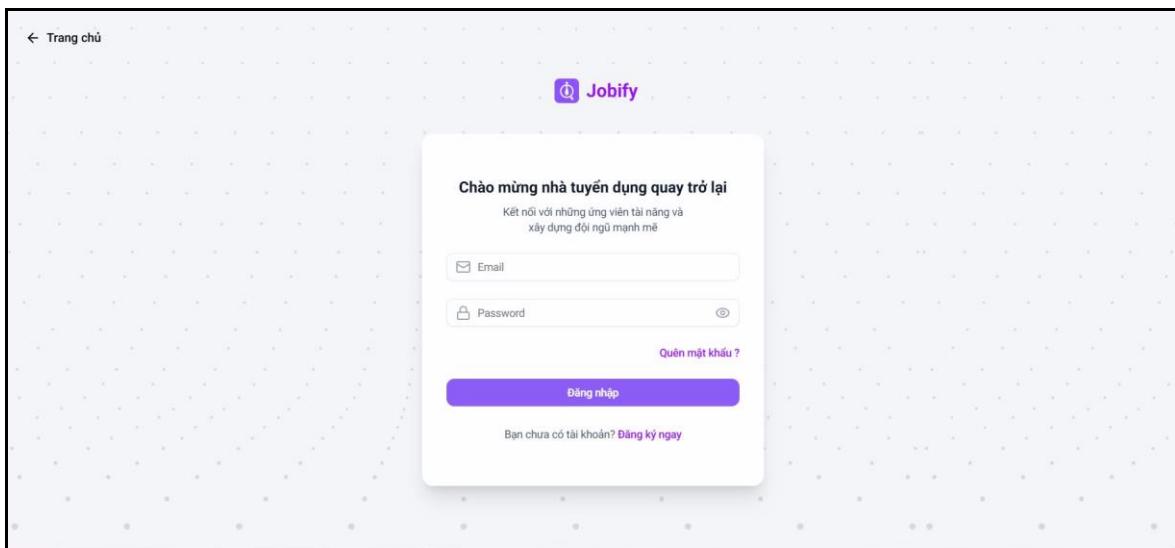
Giao diện đăng ký người tìm việc được xây dựng nhằm hỗ trợ ứng viên tạo mới tài khoản để tham gia và sử dụng các chức năng của hệ thống tìm việc. Đây là bước khởi đầu quan trọng giúp hệ thống thu thập thông tin cơ bản của người dùng, phục vụ cho quá trình xác thực, quản lý hồ sơ và hỗ trợ ứng tuyển việc làm sau này.



Hình 4.21: Giao diện trang đăng ký tài khoản người tìm việc

#### 4.2.3 Giao diện đăng nhập nhà tuyển dụng

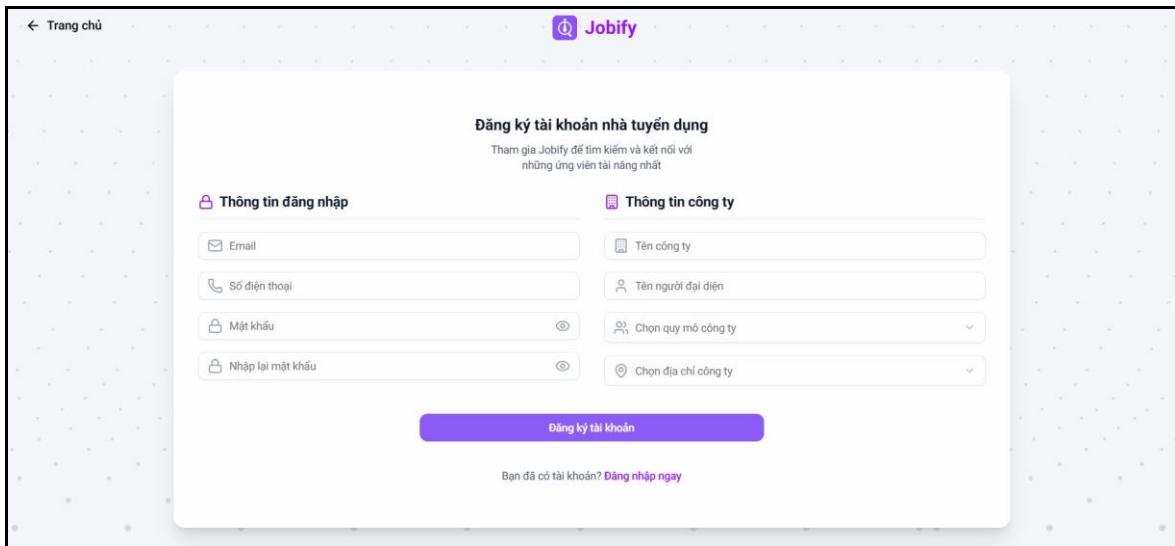
Giao diện đăng nhập nhà tuyển dụng được thiết kế nhằm xác thực thông tin tài khoản của doanh nghiệp trước khi truy cập vào các chức năng quản lý trên hệ thống tìm việc. Đây là bước quan trọng giúp đảm bảo tính bảo mật, phân quyền đúng đối tượng và bảo vệ dữ liệu tuyển dụng của doanh nghiệp.



Hình 4.22: Giao diện trang đăng nhập tài khoản nhà tuyển dụng

#### 4.2.4 Giao diện đăng ký tài khoản nhà tuyển dụng

Giao diện đăng ký tài khoản nhà tuyển dụng được xây dựng nhằm hỗ trợ doanh nghiệp tạo tài khoản để tham gia hệ thống tìm việc và sử dụng các chức năng tuyển dụng trực tuyến. Chức năng này cho phép hệ thống thu thập đầy đủ thông tin cần thiết của nhà tuyển dụng, phục vụ cho việc xác thực, quản lý doanh nghiệp và đăng tin tuyển dụng.

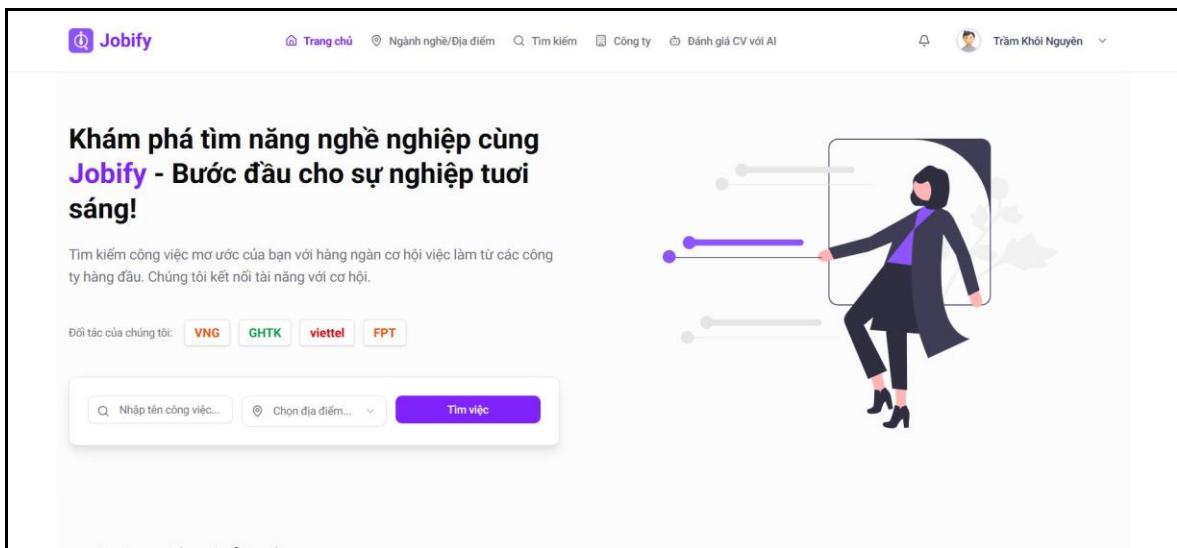


Hình 4.23: Giao diện trang đăng ký tài khoản nhà tuyển dụng

#### 4.2.5 Giao diện trang chủ

Giao diện trang chủ của hệ thống tìm việc được thiết kế nhằm giới thiệu tổng quan về website và hỗ trợ người dùng tiếp cận nhanh các chức năng chính. Tại khu vực trung tâm, hệ thống cung cấp thanh tìm kiếm cho phép người dùng tra cứu việc

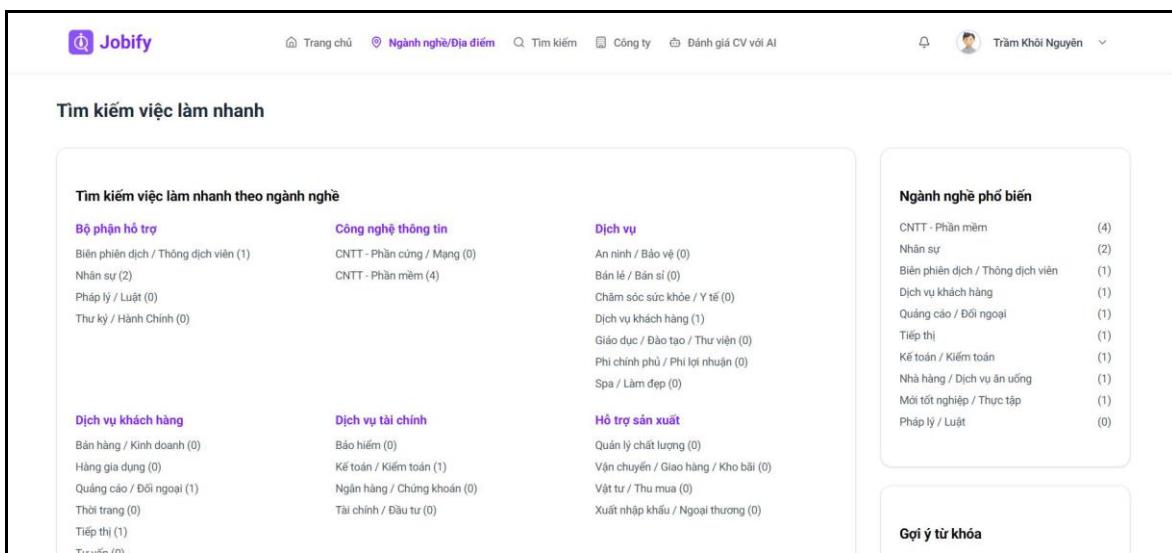
làm theo tên công việc và địa điểm, giúp rút ngắn thời gian tìm kiếm thông tin. Trang chủ hiển thị các chức năng như các ngành nghề phổ biến, các công ty và các việc làm mới nhất giúp người dùng có thể dễ dàng tìm kiếm và truy cập qua đó tăng khả năng tìm được công việc phù hợp. Đồng thời thanh điều hướng phía trên hỗ trợ truy cập nhanh đến các chức năng như tìm kiếm việc làm, danh sách công ty và đánh giá CV bằng AI.



Hình 4.24: Giao diện trang chủ

### 4.2.6 Giao diện trang ngành nghề, địa điểm

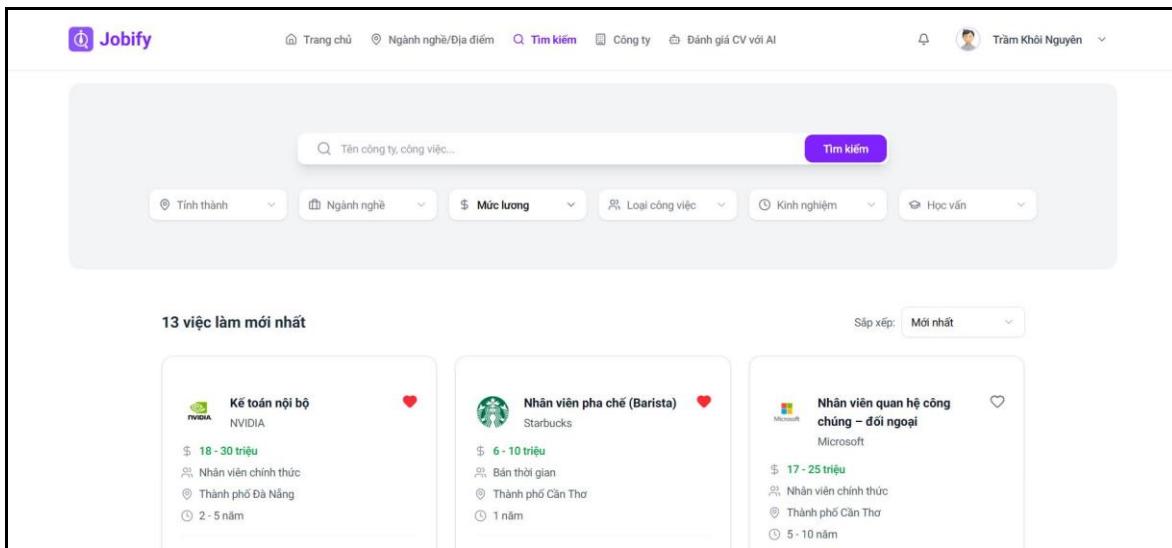
Giao diện trang ngành nghề, địa điểm được xây dựng nhằm hỗ trợ người tìm việc tra cứu và tiếp cận nhanh các cơ hội việc làm theo từng lĩnh vực ngành nghề và khu vực địa lý. Trang hiển thị danh sách các ngành nghề được phân nhóm rõ ràng, kèm theo số lượng công việc tương ứng, giúp người dùng dễ dàng định hướng nhu cầu tìm kiếm. Bên cạnh đó, hệ thống hỗ trợ lọc theo địa điểm làm việc và gợi ý các ngành nghề phổ biến, giúp người tìm việc nhanh chóng tiếp cận những lĩnh vực có nhu cầu tuyển dụng cao.



Hình 4.25: Giao diện trang ngành nghề, địa điểm

#### 4.2.7 Giao diện trang tìm kiếm việc làm

Giao diện trang tìm kiếm việc làm được thiết kế nhằm hỗ trợ người tìm việc tra cứu và lựa chọn công việc phù hợp một cách nhanh chóng và hiệu quả. Tại khu vực phía trên, hệ thống cung cấp thanh tìm kiếm theo tên công việc hoặc tên công ty, kết hợp với các bộ lọc chi tiết như tỉnh/thành phố, ngành nghề, mức lương, loại công việc, kinh nghiệm và học vấn, giúp người dùng dễ dàng thu hẹp phạm vi tìm kiếm theo nhu cầu cá nhân.

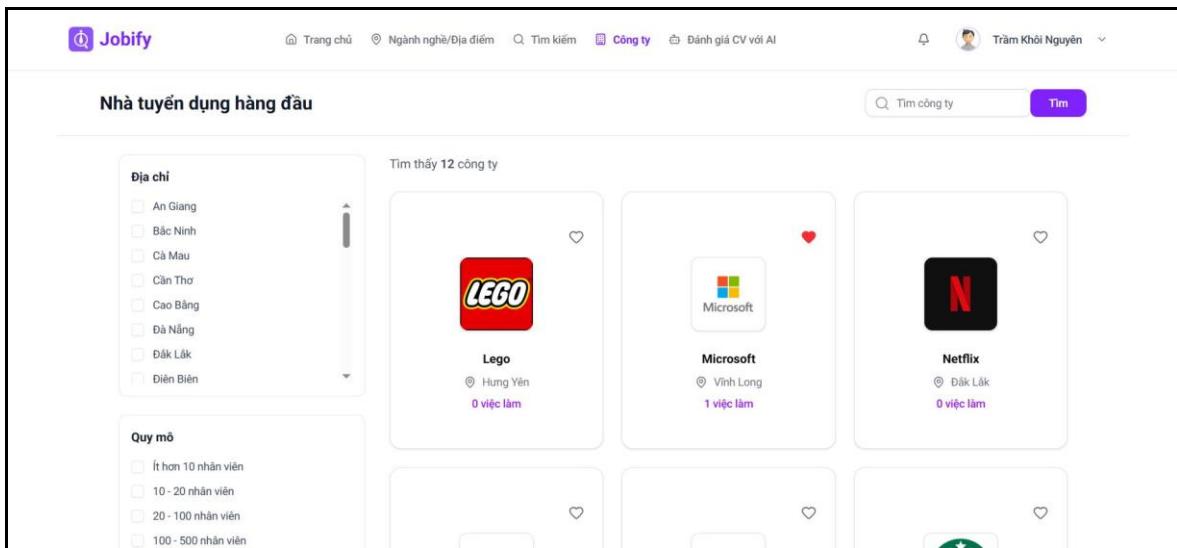


Hình 4.26: Giao diện trang tìm kiếm việc làm

#### 4.2.8 Giao diện trang tìm kiếm công ty

Giao diện trang tìm kiếm công ty được xây dựng nhằm hỗ trợ người tìm việc tra cứu và tiếp cận thông tin các nhà tuyển dụng đang hoạt động trên hệ thống tìm việc.

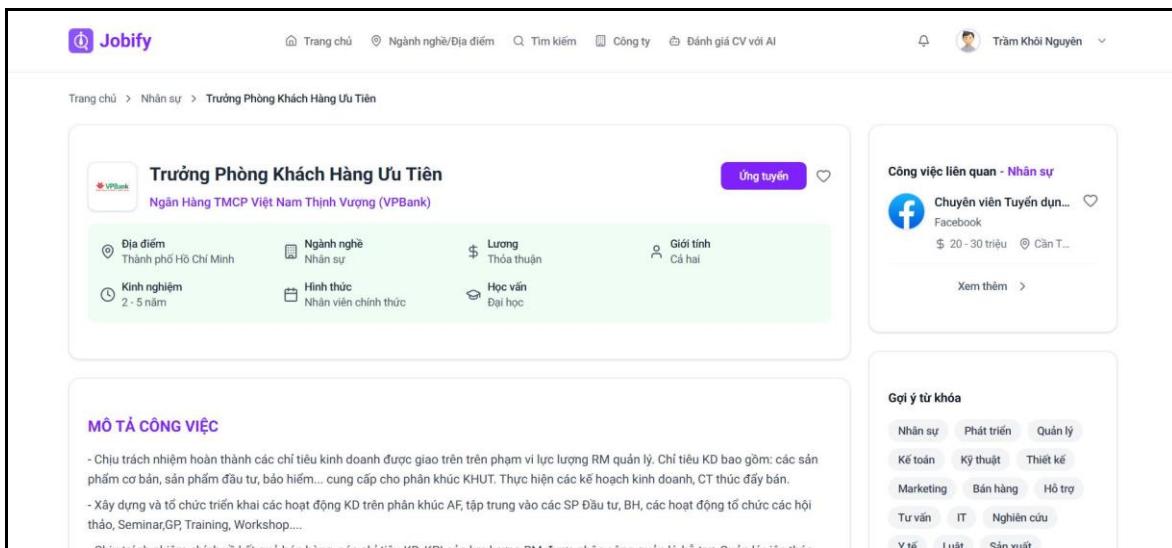
Thông qua giao diện này, người dùng có thể tìm hiểu tổng quan về doanh nghiệp, từ đó lựa chọn công ty phù hợp trước khi xem và ứng tuyển các vị trí tuyển dụng.



Hình 4.27: Giao diện trang tìm kiếm công ty

### 4.2.9 Giao diện trang chi tiết công việc

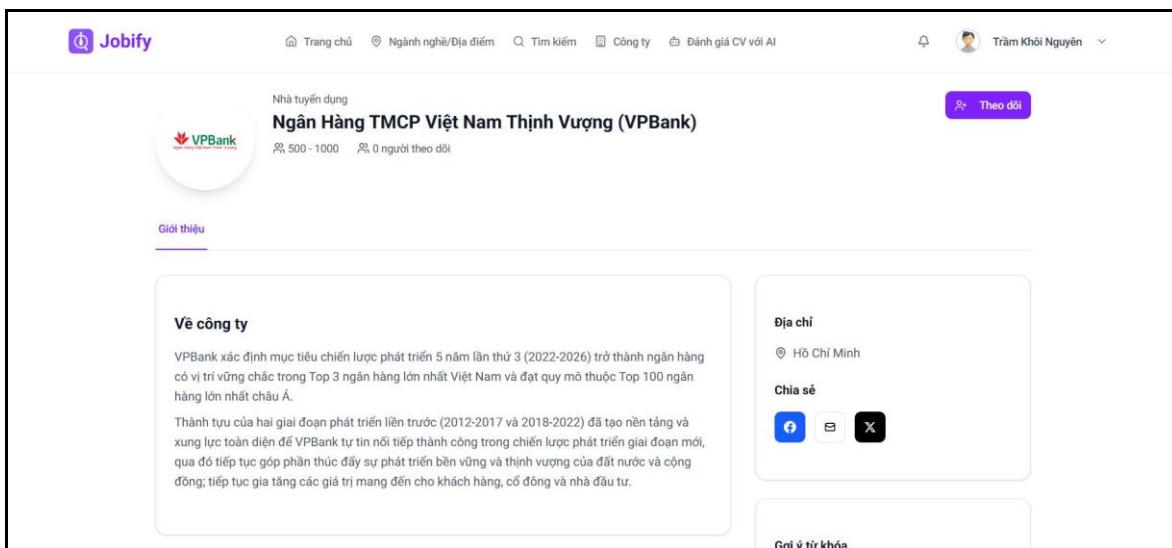
Giao diện trang chi tiết công việc cung cấp đầy đủ thông tin về một vị trí tuyển dụng cụ thể trên hệ thống tìm việc, giúp người tìm việc đánh giá mức độ phù hợp trước khi ứng tuyển. Trang hiển thị các thông tin chính như tên công việc, nhà tuyển dụng, địa điểm làm việc, mức lương, yêu cầu kinh nghiệm, hình thức làm việc và trình độ học vấn. Ngoài ra, giao diện trình bày rõ các nội dung mô tả công việc, yêu cầu công việc và thông tin khác. Hệ thống hỗ trợ chức năng ứng tuyển trực tuyến và lưu công việc yêu thích, đồng thời gợi ý các công việc liên quan nhằm mở rộng cơ hội tìm kiếm.



Hình 4.28: Giao diện trang chi tiết công việc

#### 4.2.10 Giao diện trang chi tiết nhà tuyển dụng

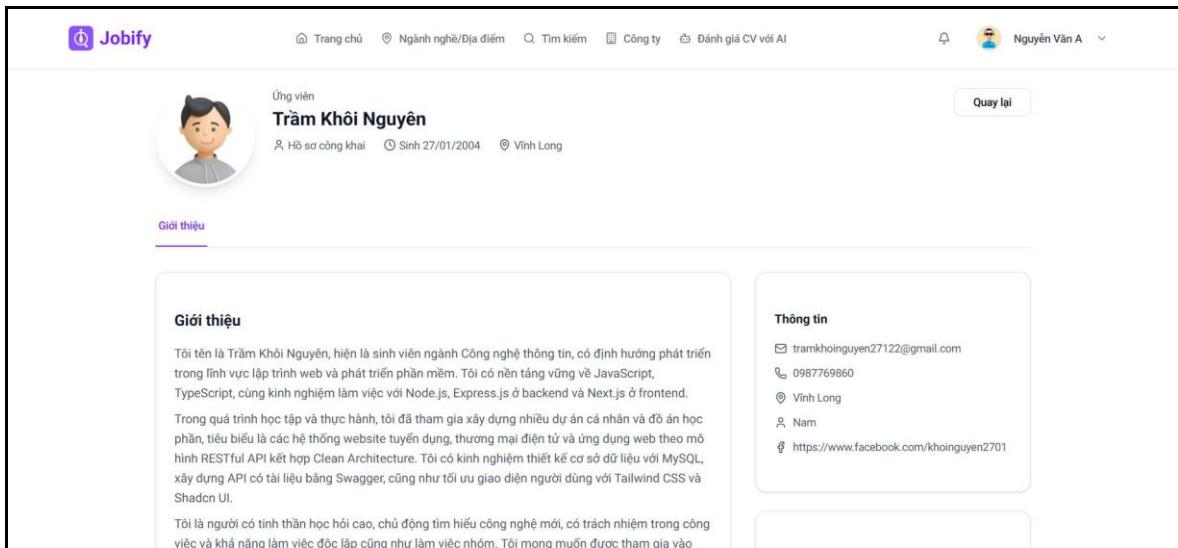
Giao diện trang chi tiết nhà tuyển dụng cung cấp thông tin tổng quan về doanh nghiệp trên hệ thống tìm việc, bao gồm tên công ty, quy mô, địa điểm hoạt động và phần giới thiệu doanh nghiệp. Đồng thời, giao diện hiển thị danh sách các công việc đang được nhà tuyển dụng đăng tuyển. Ngoài ra, hệ thống hỗ trợ chức năng theo dõi nhà tuyển dụng, giúp người tìm việc lưu lại các doanh nghiệp quan tâm để thuận tiện trong việc theo dõi và liên hệ sau này. Nội dung trang giúp người tìm việc hiểu rõ hơn về môi trường làm việc, định hướng phát triển của doanh nghiệp cũng như các cơ hội việc làm hiện có.



Hình 4.29: Giao diện trang chi tiết nhà tuyển dụng

#### 4.2.11 Giao diện trang chi tiết người tìm việc

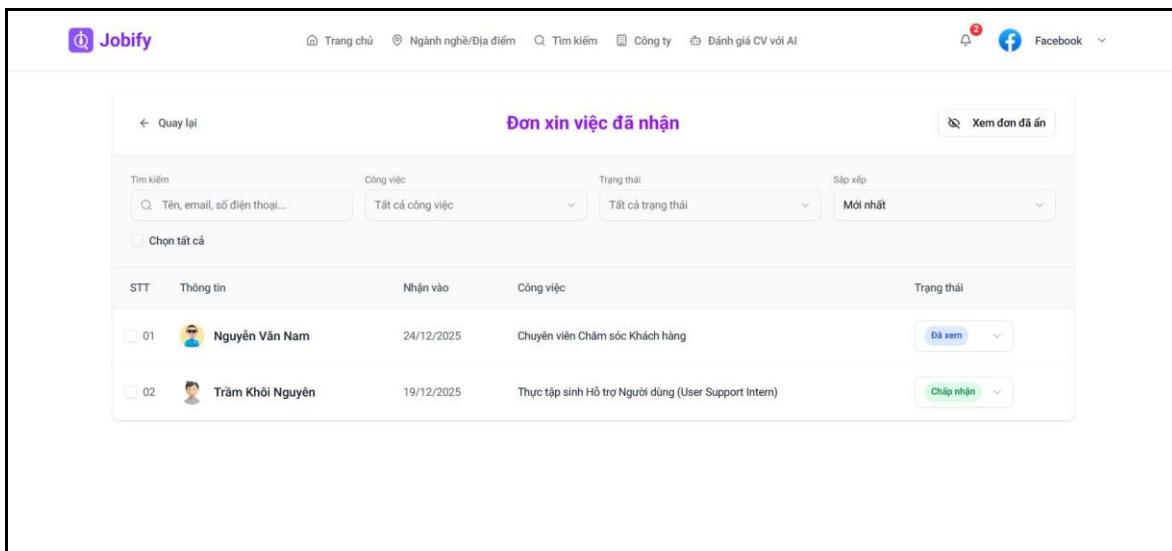
Giao diện trang chi tiết người tìm việc được thiết kế nhằm hiển thị đầy đủ và trực quan thông tin hồ sơ cá nhân của ứng viên, phục vụ cho quá trình đánh giá và tuyển chọn của nhà tuyển dụng. Trang giao diện trình bày các thông tin cơ bản như họ tên, ảnh đại diện, ngày sinh, địa điểm sinh sống và trạng thái hồ sơ. Bên cạnh đó, hệ thống cung cấp khu vực giới thiệu bản thân, giúp ứng viên mô tả chi tiết về trình độ học vấn, kỹ năng, kinh nghiệm và định hướng nghề nghiệp. Các thông tin liên hệ như email, số điện thoại và liên kết mạng xã hội cũng được hiển thị rõ ràng, hỗ trợ nhà tuyển dụng dễ dàng kết nối với ứng viên khi cần thiết.



Hình 4.30: Giao diện trang chi tiết người tìm việc

#### 4.2.12 Giao diện trang quản lý đơn ứng tuyển

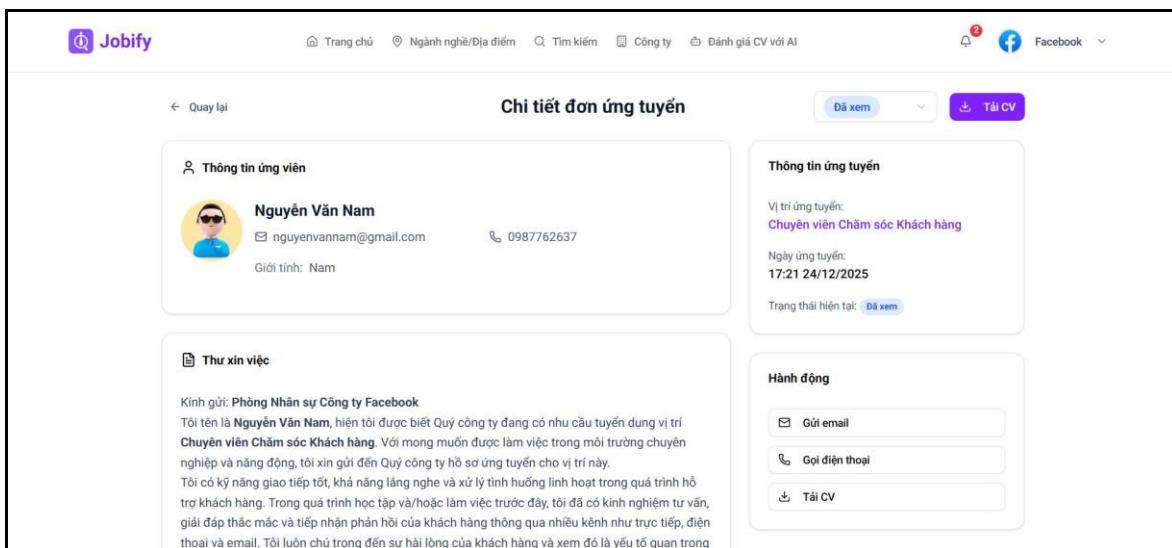
Giao diện trang quản lý đơn ứng tuyển được thiết kế nhằm hỗ trợ nhà tuyển dụng theo dõi, quản lý và xử lý các hồ sơ ứng viên đã nộp vào hệ thống. Trang hiển thị danh sách các đơn ứng tuyển dưới dạng bảng, cung cấp đầy đủ các thông tin quan trọng như tên ứng viên, ngày nộp hồ sơ, vị trí ứng tuyển và trạng thái xử lý. Hệ thống tích hợp các chức năng tìm kiếm và lọc theo tên ứng viên, công việc và trạng thái đơn, giúp nhà tuyển dụng dễ dàng tra cứu và quản lý số lượng lớn hồ sơ. Ngoài ra, nhà tuyển dụng có thể thực hiện các thao tác như xem chi tiết hồ sơ, chấp nhận hoặc từ chối đơn ứng tuyển trực tiếp trên giao diện.



Hình 4.31: Giao diện trang quản lý đơn ứng tuyển

#### 4.2.13 Giao diện trang chi tiết đơn ứng tuyển

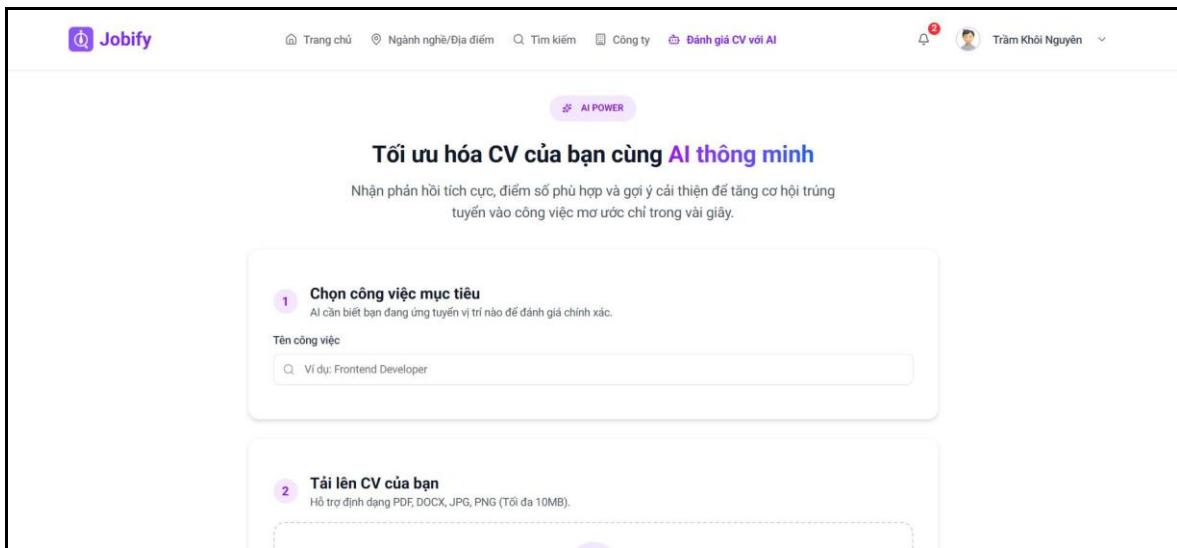
Giao diện trang chi tiết đơn ứng tuyển được xây dựng nhằm cung cấp đầy đủ thông tin liên quan đến một hồ sơ ứng tuyển cụ thể trên hệ thống tìm việc. Trang hiển thị thông tin ứng viên, vị trí ứng tuyển, thời gian nộp hồ sơ, trạng thái xử lý và nội dung thư xin việc. Bên cạnh đó, giao diện hỗ trợ các chức năng như xem và tải CV, liên hệ ứng viên qua email hoặc số điện thoại, giúp nhà tuyển dụng thuận tiện trong quá trình đánh giá và xử lý hồ sơ. Giao diện được thiết kế rõ ràng, trực quan, hỗ trợ hiệu quả cho công tác quản lý và theo dõi đơn ứng tuyển.



Hình 4.32: Giao diện trang chi tiết đơn ứng tuyển

#### 4.2.14 Giao diện trang đánh giá CV với AI

Giao diện trang đánh giá CV với AI được xây dựng nhằm hỗ trợ người tìm việc tối ưu hóa hồ sơ xin việc thông qua việc ứng dụng trí tuệ nhân tạo. Trang cho phép người dùng lựa chọn vị trí công việc mục tiêu và tải lên CV để hệ thống thực hiện phân tích, đánh giá mức độ phù hợp với yêu cầu tuyển dụng. Hệ thống cung cấp các phản hồi như điểm đánh giá, mức độ phù hợp và các gợi ý cải thiện nội dung CV, giúp người dùng nâng cao chất lượng hồ sơ và tăng khả năng trúng tuyển.



Hình 4.33: Giao diện trang đánh giá CV với AI

## CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1 Kết luận

Những kết quả đạt được:

- Xây dựng được một website tìm việc và ứng tuyển việc làm đáp ứng được yêu cầu của đề tài.
- Tìm hiểu và vận dụng các kiến thức về phát triển ứng dụng web.
- Tiếp cận và áp dụng thêm các công nghệ mới trong quá trình thực hiện đề tài.
- Tích lũy được kinh nghiệm làm việc và nghiên cứu độc lập.

Hạn chế: Do thời gian thực hiện có hạn, cùng với kinh nghiệm và kiến thức chuyên môn còn chưa cao, nên hệ thống chưa đạt được mức độ hoàn thiện và phức tạp như các ứng dụng thực tế.

Kết luận: Dù có những hạn chế, nhưng dự án đã hoàn thành đúng tiến độ. Đồng thời đảm bảo rằng tất cả các chức năng chính đều hoạt động như mong đợi. Đây là một bước quan trọng trong sự phát triển của tôi trong lĩnh vực phát triển phần mềm.

### 5.2 Hướng phát triển

- Tối ưu hóa hiệu suất và cải thiện khả năng mở rộng của hệ thống
- Tích hợp thêm trí tuệ nhân tạo để gợi ý công việc phù hợp với hồ sơ và hành vi của người tìm việc, đồng thời hỗ trợ nhà tuyển dụng trong việc sàng lọc và đánh giá ứng viên.
- Bổ sung chức năng tương tác như nhắn tin trực tiếp, thông báo theo thời gian thực cho toàn bộ chức năng và đặt lịch phỏng vấn sẽ giúp cải thiện quá trình tuyển dụng.
- Bổ sung trang dashboard cho nhà tuyển dụng để có thể quản lý công việc và ứng viên dễ dàng hơn và hỗ trợ thống kê dữ liệu trực quan.

## **DANH MỤC TÀI LIỆU THAM KHẢO**

- [1] 200Lab, "MySQL là gì? Hướng dẫn Cài đặt và Sử dụng MySQL | 200Lab Blog," [Online]. Available: <https://200lab.io/blog/mysql-la-gi>. [Accessed 27 12 2025].
- [2] 200lab, "Clean Architecture là gì - Ưu nhược và cách dùng hợp lý," [Online]. Available: <https://200lab.io/blog/clean-architecture-uu-nhuoc-va-cach-dung-hop-ly>. [Accessed 26 12 2025].
- [3] 200lab, "NextJS là gì? Kiến trúc NextJS cơ bản bạn cần biết | 200Lab Blog," [Online]. Available: <https://200lab.io/blog/nextjs-la-gi>. [Accessed 25 12 2025].
- [4] Fptshop, "Clean Architecture - cơ sở tạo nên phần mềm linh hoạt, dễ bảo trì," [Online]. Available: <https://fptshop.com.vn/tin-tuc/danh-gia/clean-architecture-180441>. [Accessed 22 12 2025].
- [5] Next.js, "What is Next.js?," [Online]. Available: <https://nextjs.org/docs#what-is-nextjs>. [Accessed 24 12 2025].
- [6] Node.js, "Node.js - Introduction to Node.js," [Online]. Available: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>. [Accessed 21 12 2025].
- [7] Restfulapi, "What is REST?: REST API Tutorial," [Online]. Available: <https://restfulapi.net>. [Accessed 22 12 2025].
- [8] TopDev, "Tìm hiểu về ExpressJS và những ưu nhược điểm của ExpressJS," [Online]. Available: <https://topdev.vn/blog/uu-nhuoc-diem-cua-expressjs>. [Accessed 23 12 2025].
- [9] Viblo, "Khám Phá Các Ưu Điểm của Next.js trong Phát Triển Web - Viblo," [Online]. Available: <https://viblo.asia/p/kham-pha-cac-uu-diem-cua-nextjs-trong-phat-trien-web-5pPLk9Re4RZ>. [Accessed 25 12 2025].