

**BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



HỆ THỐNG QUẢN LÝ WEB TÌM VIỆC

Sinh viên thực hiện

1. Đặng Thị Quyền Cơ – 20043001

Phục lục

CHƯƠNG 1: GIỚI THIỆU	2
1.1 Tổng quan	2
1.2 Mục tiêu đề tài	2
1.3 Phạm vi	2
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	4
2.1 Kiến trúc phần mềm	4
2.1.1 Kiến trúc phần mềm là gì?	4
2.1.2 Mô hình phân lớp (Layered Architecture)	4
2.2 Các công nghệ được sử dụng	6
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG	8
3.1 Tổng quan kiến trúc hệ thống	8
3.1.1 Lựa chọn kiến trúc phần mềm	8
3.1.1.1 Lý do lựa chọn	8
3.1.1.2 Trade off	9
3.1.2 Sơ đồ kiến trúc	10
3.2 Sơ đồ CSDL	12
3.3 Sơ đồ lớp	13
CHƯƠNG 4: TRIỂN KHAI DỰ ÁN	14
4.1 Công cụ sử dụng	14
4.2 Mô tả chức năng	14
4.3 Giao diện màn hình	15
CHƯƠNG 5: NHỮNG KHÓ KHĂN	22

CHƯƠNG 1: GIỚI THIỆU

1.1 Tổng quan

Ngày nay, nhu cầu kết nối giữa nhà tuyển dụng và người tìm việc đang gia tăng mạnh mẽ nhờ vào sự phát triển của công nghệ. Các nền tảng tìm việc đóng vai trò cầu nối quan trọng, giúp tiết kiệm thời gian và công sức cho cả hai bên. Tuy nhiên, nhiều nền tảng hiện nay vẫn còn hạn chế về giao diện thân thiện hoặc thiếu tính năng tùy chỉnh phù hợp với thị trường nội địa. Vì vậy, dự án xây dựng một **website tìm việc** hiện đại, dễ sử dụng và tích hợp các tính năng cần thiết là một hướng đi tiềm năng, đáp ứng yêu cầu thực tế.

1.2 Mục tiêu đề tài

Áp dụng Spring Framework: Tích hợp Spring Boot để phát triển backend với kiến trúc RESTful API.

Xây dựng kết nối giữa frontend và backend. Triển khai giao tiếp API giữa giao diện người dùng (frontend) và máy chủ (backend).

Phát triển hệ thống web tìm việc hoàn chỉnh: Bao gồm các tính năng cơ bản phục vụ người tìm việc, nhà tuyển dụng, và quản trị viên.

Nâng cao kiến thức thực hành, tăng cường kỹ năng sử dụng Spring Framework, triển khai API và giao tiếp giữa các thành phần của hệ thống.

1.3 Phạm vi

1.3.1 Đối tượng sử dụng:

- Ứng viên: Người tìm việc muốn tìm và ứng tuyển công việc.

- Nhà tuyển dụng: Doanh nghiệp đăng tin tuyển dụng và quản lý ứng viên.
- Quản trị viên: Quản lý nội dung và dữ liệu hệ thống.

1.3.2 Chức năng chính:

- **Ứng viên:**
 - Đăng ký, đăng nhập, và quản lý hồ sơ cá nhân.
 - Tìm kiếm công việc
- **Nhà tuyển dụng:**
 - Đăng ký, đăng nhập và quản lý hồ sơ công ty
 - Tạo tài khoản công ty và quản lý thông tin tuyển dụng.
 - Đăng tin tuyển dụng và xem danh sách ứng viên.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Kiến trúc phần mềm

2.1.1 Kiến trúc phần mềm là gì?

Theo Wikipedia, kiến trúc phần mềm (Software architecture) đề cập đến cấu trúc cơ bản của một hệ thống phần mềm và quy tắc của việc tạo ra những cấu trúc và hệ thống như vậy. Mỗi cấu trúc bao gồm sự sắp xếp của các yếu tố phần mềm, mối quan hệ giữa các yếu tố, và tính chất của các yếu tố đó.

“Software architecture is the set of structures needed to reason about a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.”

Thiết kế kiến trúc phần mềm (software architecture design) ản dụ cho việc thiết kế một toà nhà. Nó được coi như bản thiết kế của hệ thống phần mềm. Từ đó có thể hình dung ra lượng công việc cụ thể phải làm, dẫn tới, việc quản lý dự án trở nên dễ dàng hơn.

Cho tới giờ vẫn chưa có một định nghĩa phản ánh đầy đủ nhất về kiến trúc phần mềm bởi hai lý do:

- Sự phát triển nhanh chóng của công nghệ, môi trường phát triển, các loại mô hình kiến trúc phần mềm khác nhau khiến cho các định nghĩa trước đó nhanh chóng trở nên lỗi thời.
- Sự khác biệt về quan điểm giữa các trường phái. Ví dụ: một số software architect coi kiến trúc phần mềm là bản thiết kế chi tiết của hệ thống, trong khi những người khác định nghĩa nó là lộ trình phát triển hệ thống.

2.1.2 Mô hình phân lớp (Layered Architecture)

Hiện nay, có rất nhiều mô hình kiến trúc phần mềm khác nhau, mỗi mô hình đều có những ưu điểm, nhược điểm riêng phù hợp cho từng loại ứng dụng.

Mô hình lớp (Layered Architecture) là một kiểu kiến trúc phần mềm trong đó hệ thống được chia thành các lớp (layer) theo chức năng. Mỗi lớp thực hiện một vai trò cụ thể và giao tiếp với các lớp lân cận theo một cách có tổ chức và có thứ tự. Các lớp thường được thiết kế theo mô hình phân cấp, trong đó mỗi lớp chỉ phụ thuộc vào lớp bên dưới nó và cung cấp các dịch vụ cho lớp trên.

2.1.2.1 Nguyên tắc hoạt động của mô hình lớp

Phân chia lớp: Hệ thống được chia thành các lớp riêng biệt dựa trên chức năng, ví dụ như:

- Lớp trình bày (Presentation Layer): Quản lý giao diện người dùng và xử lý đầu vào từ người dùng.
- Lớp nghiệp vụ (Business Logic Layer): Xử lý logic nghiệp vụ và các quy tắc kinh doanh.
- Lớp dữ liệu (Data Access Layer): Quản lý việc truy cập và lưu trữ dữ liệu.
- Lớp cơ sở hạ tầng (Infrastructure Layer): Cung cấp các dịch vụ chung như logging, giao tiếp mạng, v.v.

Giao tiếp giữa các lớp: Các lớp chỉ giao tiếp với lớp ngay bên dưới và bên trên nó. Dữ liệu và yêu cầu di chuyển từ lớp này sang lớp khác theo một hướng cụ thể, thường từ lớp trên xuống dưới (top-down) hoặc từ dưới lên trên (bottom-up).

Đóng gói và trừu tượng hóa: Mỗi lớp đóng gói chức năng của nó và cung cấp các giao diện rõ ràng để các lớp khác tương tác mà không cần biết chi tiết triển khai bên trong.

2.1.2.2 Ưu điểm của mô hình lớp

Dễ hiểu và dễ quản lý:

- Cấu trúc phân lớp giúp các nhà phát triển dễ dàng hiểu và quản lý hệ thống.
- Mỗi lớp có thể được phát triển và duy trì độc lập.

Tăng khả năng tái sử dụng:

- Các lớp có thể được tái sử dụng trong các hệ thống khác nếu chúng cung cấp các chức năng chung.
- Logic nghiệp vụ có thể được sử dụng lại trong nhiều ứng dụng khác nhau.

Tăng khả năng bảo trì:

- Khi có thay đổi trong một lớp, chỉ cần thay đổi lớp đó mà không ảnh hưởng đến các lớp khác.
- Giảm sự phụ thuộc và tăng tính mô-đun của hệ thống.

Tách biệt các mối quan tâm:

- Mỗi lớp giải quyết một khía cạnh cụ thể của hệ thống, giúp tách biệt rõ ràng các mối quan tâm (separation of concerns).

*2.1.2.3 Nhược điểm của mô hình lớp***Hiệu suất:**

- Hệ thống có thể bị chậm do phải đi qua nhiều lớp để hoàn thành một tác vụ.
- Quá trình truyền tải dữ liệu giữa các lớp có thể tạo ra độ trễ.

Khó khăn trong việc quản lý thay đổi lớn:

- Khi có thay đổi lớn trong yêu cầu nghiệp vụ, có thể phải thay đổi nhiều lớp, gây khó khăn và tốn kém.

Cấu trúc cứng nhắc:

- Các lớp cố định và phụ thuộc lẫn nhau có thể làm hệ thống trở nên cứng nhắc, khó thích ứng với các thay đổi không mong đợi.

Tiềm năng lãng phí tài nguyên:

Mỗi lớp có thể lặp lại các thao tác đã được thực hiện ở lớp khác, dẫn đến lãng phí tài nguyên và thời gian.

2.2 Các công nghệ được sử dụng

Frontend:

- Framework: React.js
- Công cụ hỗ trợ: Axios để kết nối với backend.

Backend:

- Framework: Spring Boot (Java).
- Mô hình phát triển: RESTful API để giao tiếp với frontend.

Cơ sở dữ liệu:

- Hệ quản trị: MariaDB
- ORM: Hibernate (tích hợp với JPA).

Công cụ triển khai:

- IDE: IntelliJ IDEA.
- Build tool: Gradle.

Công cụ kiểm thử:

- Công cụ kiểm thử REST API: Postman

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1 Tổng quan kiến trúc hệ thống

3.1.1 Lựa chọn kiến trúc phần mềm

Với yêu cầu quản lý web tìm việc như trên, lựa chọn mô hình kiến trúc *Layer* là phù hợp vì tính phân tầng của hệ thống, cho phép phân chia logic ứng dụng thành các lớp riêng biệt, dễ dàng quản lý và bảo trì..

3.1.1.1 Lý do lựa chọn

- **Tách biệt rõ ràng các chức năng:** Mô hình Layer giúp tách biệt các chức năng và trách nhiệm của từng phần trong hệ thống. Mỗi layer đảm nhận một vai trò cụ thể, chẳng hạn như giao diện người dùng, logic nghiệp vụ, truy cập dữ liệu, và tích hợp với các hệ thống bên ngoài. Điều này giúp tăng tính modular và dễ bảo trì cho hệ thống.
- **Dễ dàng bảo trì và mở rộng:** Khi có thay đổi hoặc cần thêm chức năng mới, chỉ cần cập nhật trong layer tương ứng mà không ảnh hưởng đến các layer khác. Ví dụ, nếu cần thay đổi giao diện người dùng, chỉ cần chỉnh sửa tầng Presentation mà không ảnh hưởng đến tầng Business Logic hay Data Access.
- **Tăng tính linh hoạt:** Các tầng được thiết kế để độc lập với nhau, cho phép sử dụng lại các thành phần ở các dự án khác hoặc thay thế chúng mà không cần phải thay đổi toàn bộ hệ thống. Ví dụ, có thể thay thế hệ thống gửi email mà không cần thay đổi tầng Business Logic.
- **Quản lý tốt hơn các phụ thuộc:** Mô hình Layer giúp quản lý các phụ thuộc giữa các thành phần của hệ thống một cách hiệu quả. Các thành phần ở tầng cao hơn chỉ phụ thuộc vào các thành phần ở tầng thấp hơn, giúp giảm thiểu sự phức tạp trong quản lý phụ thuộc.
- **Tăng tính bảo mật:** Bằng cách tách biệt các tầng, hệ thống có thể áp dụng các biện pháp bảo mật khác nhau cho từng tầng. Ví dụ, tầng Data Access có thể có

các biện pháp bảo mật để đảm bảo dữ liệu không bị truy cập trái phép, trong khi tầng Presentation có thể áp dụng các biện pháp để ngăn chặn các cuộc tấn công từ phía người dùng.

- **Đơn giản hóa việc kiểm thử:** Từng tầng có thể được kiểm thử một cách độc lập, giúp phát hiện lỗi sớm và dễ dàng hơn. Unit test có thể được thực hiện cho tầng Business Logic mà không cần phụ thuộc vào tầng Presentation hoặc Data Access.
- **Tăng hiệu suất làm việc nhóm:** Khi phát triển một hệ thống lớn, mô hình Layer cho phép phân chia công việc một cách hiệu quả giữa các nhóm phát triển. Mỗi nhóm có thể tập trung vào một tầng cụ thể mà không cần lo lắng về các tầng khác, từ đó tăng năng suất làm việc và giảm xung đột trong quá trình phát triển.

3.1.1.2 Trade off

1. Hiệu suất

- Overhead từ việc tách lớp: Việc tách biệt các tầng có thể dẫn đến việc gọi hàm qua lại nhiều lần giữa các tầng, tạo ra overhead và có thể làm giảm hiệu suất của hệ thống. Điều này đặc biệt đáng chú ý trong các ứng dụng yêu cầu hiệu suất cao hoặc thời gian phản hồi nhanh.
- Latency tăng lên: Khi dữ liệu phải di chuyển qua nhiều tầng, độ trễ trong quá trình xử lý và phản hồi yêu cầu của người dùng có thể tăng lên.

2. Phức tạp trong thiết kế và triển khai

- Tăng độ phức tạp ban đầu: Thiết kế và triển khai một hệ thống theo kiến trúc Layer có thể phức tạp hơn so với các mô hình đơn giản khác. Đòi hỏi phải có sự hiểu biết tốt về các nguyên tắc kiến trúc và quản lý tầng.
- Tăng cường phối hợp nhóm: Việc phân chia hệ thống thành nhiều tầng đòi hỏi sự phối hợp chặt chẽ giữa các nhóm phát triển, đặc biệt trong các dự án lớn.

3. Cứng nhắc

- Giới hạn tính linh hoạt trong một số trường hợp: Mô hình Layer có thể trở nên cứng nhắc khi cần thay đổi hoặc mở rộng một số chức năng mà không ảnh hưởng đến các

tầng khác. Điều này có thể làm cho việc điều chỉnh hệ thống trở nên khó khăn và tốn thời gian.

- Quy tắc phụ thuộc nghiêm ngặt: Các tầng phải tuân thủ quy tắc phụ thuộc nghiêm ngặt, tầng trên không được phép gọi trực tiếp tầng dưới. Điều này có thể làm tăng sự phức tạp khi có yêu cầu đặc biệt hoặc khẩn cấp.

4. Khó kiểm thử tích hợp

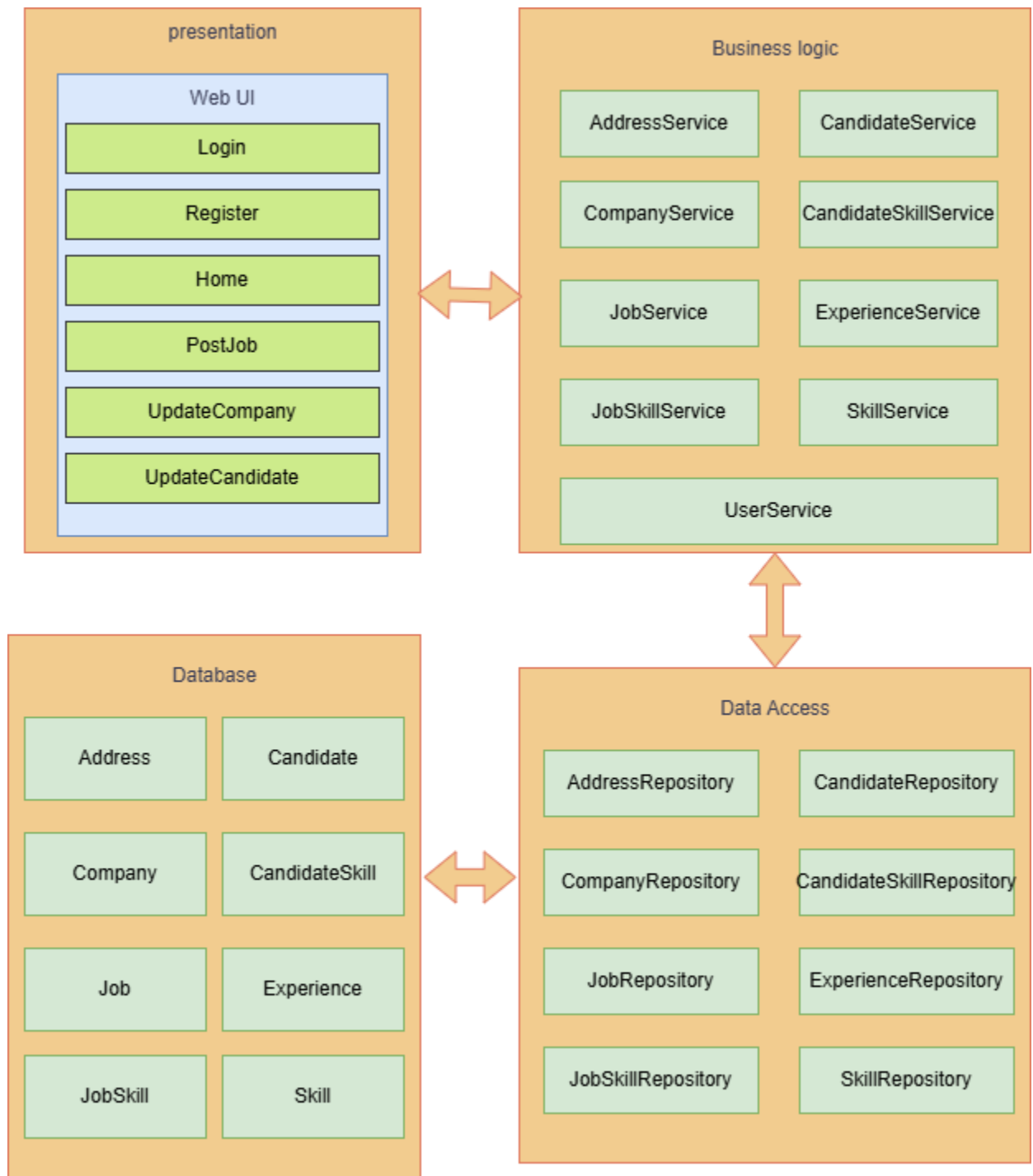
- Kiểm thử tích hợp phức tạp: Mặc dù các tầng có thể được kiểm thử độc lập, nhưng kiểm thử tích hợp giữa các tầng có thể trở nên phức tạp và tốn nhiều công sức. Việc đảm bảo các tầng hoạt động chính xác và tương tác tốt với nhau có thể đòi hỏi nhiều thời gian và tài nguyên.

5. Duy trì và cập nhật

- Khó khăn khi duy trì tính nhất quán: Khi hệ thống phát triển, việc duy trì tính nhất quán giữa các tầng có thể trở nên khó khăn. Các thay đổi nhỏ ở tầng dưới có thể yêu cầu cập nhật và điều chỉnh ở các tầng trên.
- Phụ thuộc vào kiến trúc sư phần mềm: Thiết kế và bảo trì kiến trúc Layer yêu cầu các kiến trúc sư phần mềm có kinh nghiệm và hiểu biết sâu về kiến trúc phần mềm, điều này có thể không phải lúc nào cũng có sẵn.

3.1.2 Sơ đồ kiến trúc

Để thiết kế một hệ thống quản lý tìm việc chỉ theo kiến trúc layer (tầng), chúng ta cần chia hệ thống thành các tầng khác nhau, mỗi tầng sẽ đảm nhận một số chức năng cụ thể.



Hình 1: Mô hình phân lớp

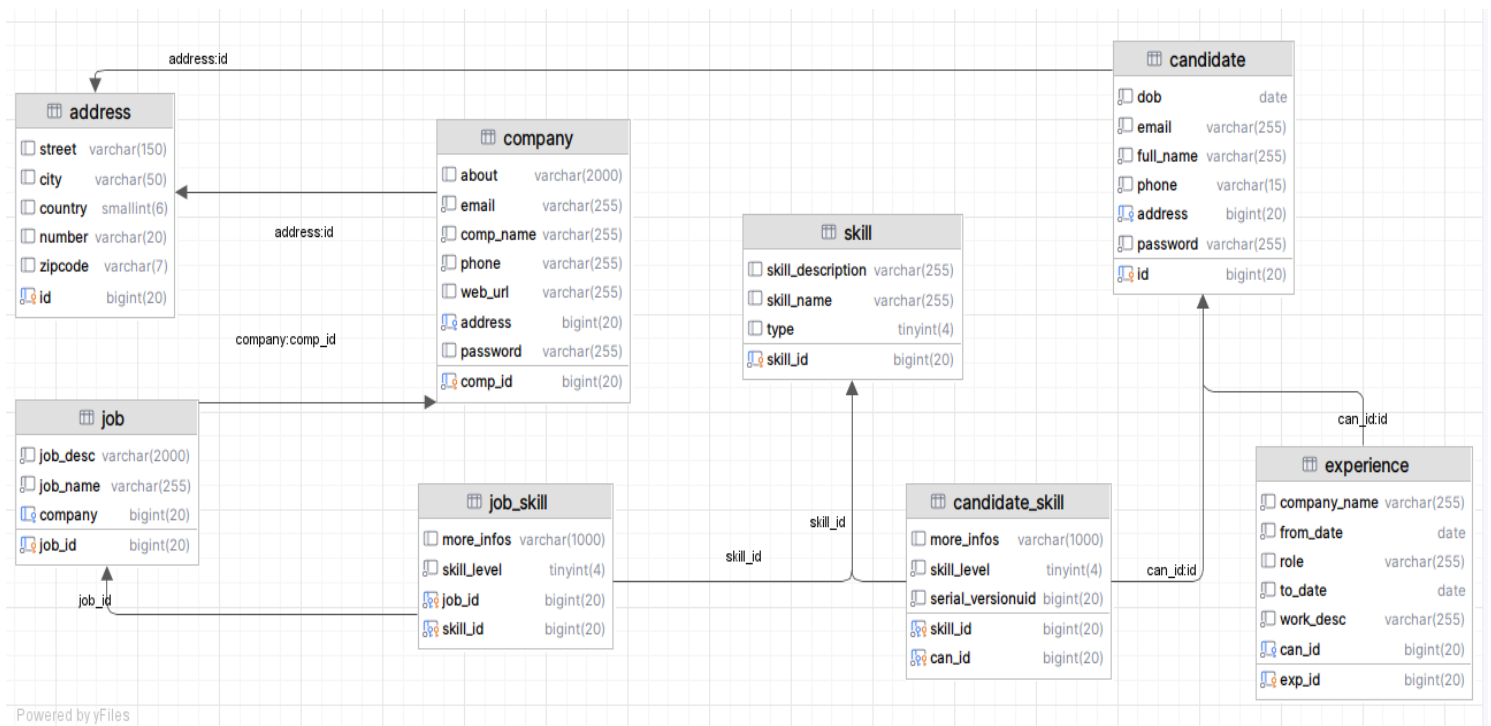
Tầng Presentation (Giao diện người dùng): chứa các giao diện người dùng Web UI:

Tầng Business Logic (Quy tắc nghiệp vụ): chứa các service giúp xử lý các quy tắc như phân trang, đăng ký, đăng nhập, tìm việc,...

Tầng Data Access (Truy cập dữ liệu): Giao tiếp với CSDL để lưu trữ và truy xuất dữ liệu

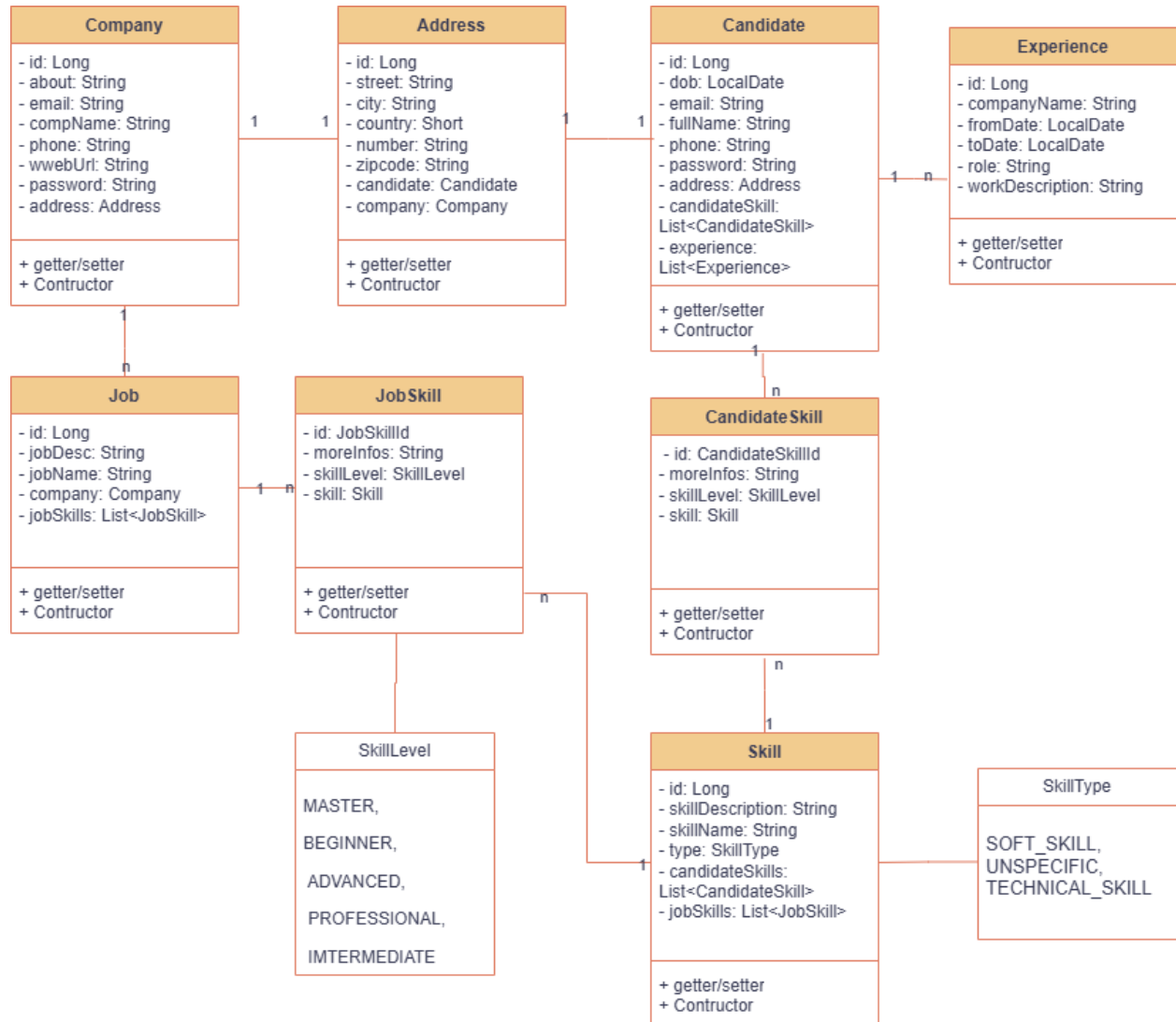
Tầng Database (Cơ sở dữ liệu): Lưu trữ cơ sở dữ liệu

3.2 Sơ đồ CSDL



Hình 2: Sơ đồ cơ sở dữ liệu

3.3 Sơ đồ lớp



Hình 3: Sơ đồ class

CHƯƠNG 4: TRIỂN KHAI DỰ ÁN

4.1 Công cụ sử dụng

Để xây dựng và triển khai hệ thống, dự án sử dụng các công cụ sau:

Frontend

- Ngôn ngữ lập trình: JavaScript, CSS.
- Framework: React.js (hỗ trợ xây dựng giao diện hiện đại).
- Thư viện hỗ trợ: Axios (kết nối API từ backend).
- Công cụ phát triển:
 - Visual Studio Code: IDE để viết mã frontend.
 - Node.js: Môi trường chạy và quản lý các gói npm.

Backend

- Ngôn ngữ lập trình: Java.
- Framework: Spring Boot (hỗ trợ xây dựng API RESTful).
- Công cụ quản lý phụ thuộc: Gradle.
- Công cụ hỗ trợ phát triển:
 - IntelliJ IDEA: IDE cho việc lập trình backend.
 - Postman: Kiểm thử các API đã triển khai.

Database

- Hệ quản trị cơ sở dữ liệu: MariaDB.

Công cụ khác

- Git: Quản lý mã nguồn và làm việc nhóm.
- Trình duyệt: Google Chrome để kiểm tra giao diện.

4.2 Mô tả chức năng

Dự án website tìm việc được chia thành các chức năng chính như sau:

Chức năng cho ứng viên

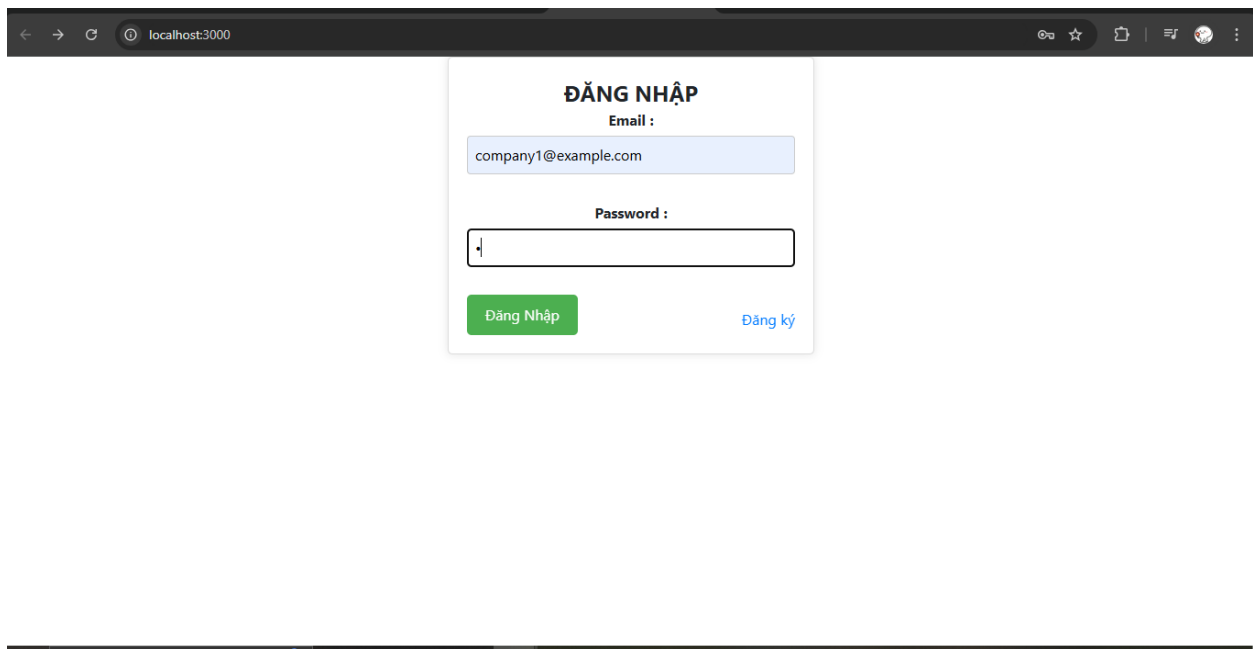
- **Đăng ký và đăng nhập:**
 - Ứng viên có thể tạo tài khoản, đăng nhập vào hệ thống bằng email và mật khẩu.
- **Quản lý hồ sơ cá nhân:**
 - Ứng viên có thể cập nhật thông tin cá nhân (tên, kỹ năng, kinh nghiệm).
 - Hồ sơ được lưu trong cơ sở dữ liệu và có thể chỉnh sửa.
- **Tìm kiếm công việc:**
 - Ứng viên có thể tìm kiếm công việc dựa trên các tiêu chí như vị trí, ngành nghề, và mức lương.
 - Kết quả trả về qua API từ backend.

Chức năng cho nhà tuyển dụng

- **Đăng ký và quản lý tài khoản công ty:**
 - Nhà tuyển dụng có thể đăng ký tài khoản và cập nhật thông tin công ty.
- **Đăng tin tuyển dụng:**
 - Nhà tuyển dụng tạo bài đăng tuyển dụng với các thông tin như: vị trí cần tuyển, yêu cầu, mức lương.
 - Tin tuyển dụng được lưu vào database và hiển thị trên trang tìm việc.

4.3 Giao diện màn hình

Giao diện đăng nhập: Nhập đúng mail và mật khẩu sẽ chuyển hướng sang trang Home, nếu sai sẽ thông báo.

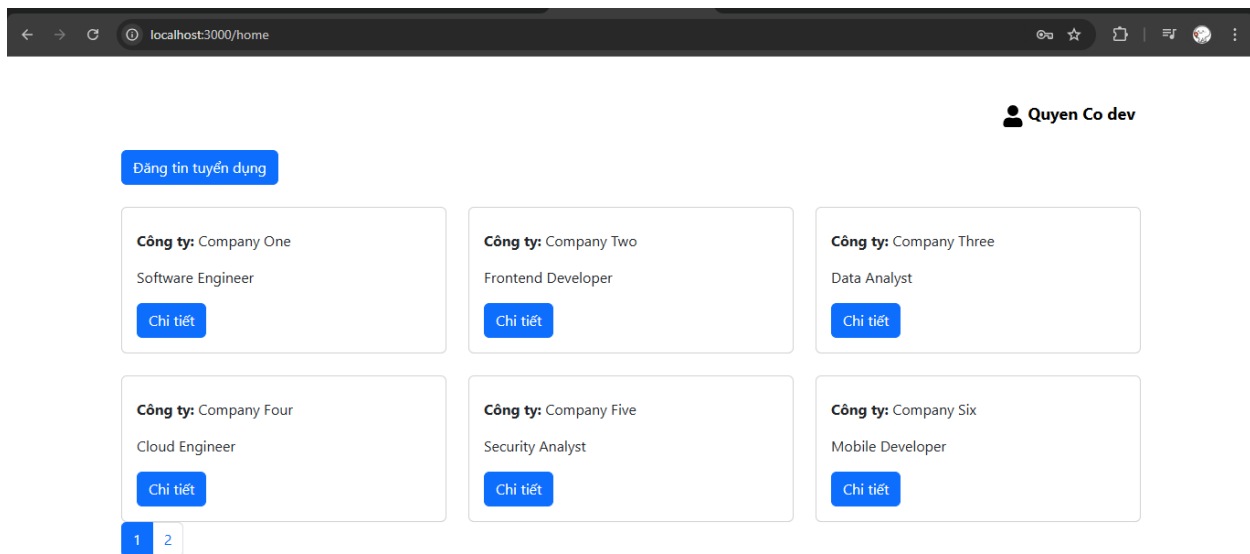


Hình 4: Giao diện đăng nhập

Giao diện đăng ký: tạo tài khoản Candidate hoặc Company bằng email và mật khẩu.

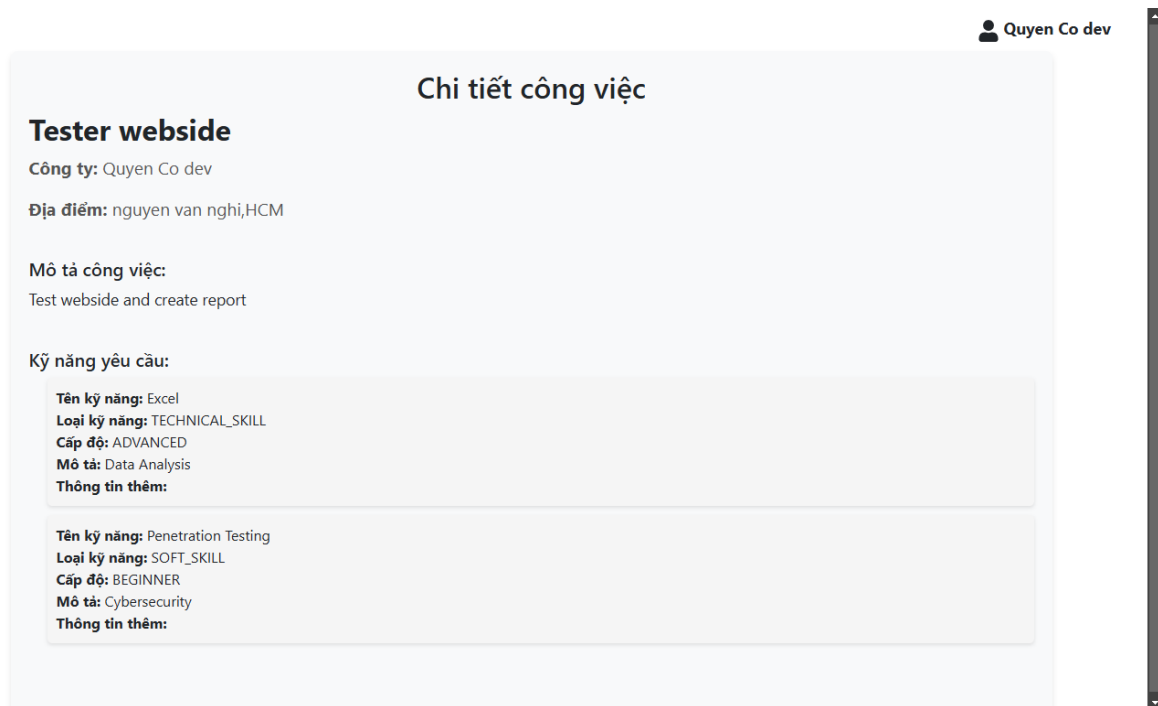
Hình 5: Giao diện trang đăng ký

Giao diện Home: khi đăng nhập thành công sẽ hiển thị trang chứa các công việc đã được đăng, nếu company đăng nhập thì sẽ hiển thị 1 button đăng tin tuyển dụng.



Hình 6: Giao diện trang chủ

Trang chi tiết tin tuyển dụng: Nhấp vào chi tiết sẽ hiển thị thông tin của công việc



Hình 7: Giao diện chi tiết tuyển dụng

Trang đăng tin tuyển dụng: tạo 1 tin tuyển dụng với các thông tin: tên công việc, mô tả công việc, có thể chọn 0 hoặc nhiều kỹ năng dựa trên các kỹ năng có sẵn, ngoài ra có nút thêm kỹ năng mới. Sau khi chọn cấp độ kỹ năng, hoặc xóa kỹ năng đã chọn.

Để thoát khỏi chế độ toàn màn hình, hãy nhấn và giữ

Thoát

Quyên Co dev

Đăng Tin Tuyển Dụng

Tên công việc

Website development

Mô tả công việc

execute the code in java

Kỹ năng

HTML/CSS

[Thêm kỹ năng mới](#)

Tên kỹ năng	Cấp độ kỹ năng	Thêm thông tin	Xóa
Java	BEGINNER	<div>Thêm thông tin chi tiết</div>	<div>Xóa</div>
HTML/CSS	BEGINNER	<div>Thêm thông tin chi tiết</div>	<div>Xóa</div>

Đăng Công Việc

Hình 8: Giao diện đăng tin tuyển dụng

Thêm kỹ năng mới: nhập các thông tin như tên kỹ năng, mô tả và cấp độ sau đó nhấn thêm.

Thêm Kỹ Năng Mới

Tên kỹ năng

Tên kỹ năng

Mô tả

Mô tả kỹ năng

Loại kỹ năng

Chọn loại kỹ năng

Lưu

Hình 9: Giao diện thêm kỹ năng

Trang cập nhật thông tin công ty: cập nhật các thông tin của công ty và địa chỉ của công ty.

 **Quyên Co dev**

Cập nhật thông tin công ty

Tên công ty

Email

Số điện thoại

Website

Mô tả công ty


Địa chỉ

Địa chỉ (Street)

Thành phố

Hình 10: Giao diện cập nhật thông tin công ty

Trang cập nhật thông tin candidate: chứa các ô cập nhật thông tin candidate và địa chỉ của candidate.



Cập nhật thông tin ứng viên

Cập nhật thông tin công ty thành công!

Họ và tên

Email

Số điện thoại

Ngày sinh

Địa chỉ

Địa chỉ (Street)

Hình 11: Giao diện cập nhật thông tin ứng viên

CHƯƠNG 5: NHỮNG KHÓ KHĂN

5.1 Về Kỹ Thuật

Kết nối Frontend và Backend:

- Việc thiết lập giao tiếp giữa React.js và Spring Boot thông qua RESTful API đòi hỏi sự hiểu biết sâu về cấu trúc dữ liệu, định dạng JSON, và cơ chế xử lý HTTP request/response.
- Một số lỗi thường gặp như CORS (Cross-Origin Resource Sharing) đã gây mất thời gian để xử lý.

Quản lý cơ sở dữ liệu:

- Thiết kế và tối ưu cấu trúc cơ sở dữ liệu ban đầu gặp khó khăn do thiếu kinh nghiệm trong việc xây dựng các bảng liên kết phức tạp.
- Đảm bảo tính nhất quán và an toàn dữ liệu khi nhiều người dùng truy cập đồng thời cũng là một thách thức.

Sử dụng công nghệ mới:

- Spring Boot là công nghệ mới đối với nhóm, cần thời gian để tìm hiểu cách cấu hình, triển khai và tích hợp với các thành phần khác.
- React.js tuy dễ tiếp cận nhưng việc quản lý trạng thái (state) trong ứng dụng lớn cũng đòi hỏi học tập và nghiên cứu thêm.

5.2 Về Quản Lý Thời Gian

Phân chia công việc:

- Ban đầu, nhóm gặp khó khăn trong việc phân chia công việc hợp lý, dẫn đến một số thành viên bị quá tải trong khi các thành viên khác không nắm rõ nhiệm vụ.

Thời gian hạn chế:

- Do phải kết hợp giữa học tập và làm dự án, nhóm khó cân đối thời gian để thực hiện các chức năng phức tạp.