**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: NGHIÊN CỨU VÀ ỨNG DỤNG CÔNG CỤ OR-TOOLS**

**VÀO XÂY DỰNG HỆ THỐNG ĐIỀU PHỐI NGUỒN LỰC TỰ ĐỘNG**

Giảng viên hướng dẫn: THS. TRẦN PHONG NHÃ

Sinh viên thực hiện: NGUYỄN HOÀNG PHÁT

Lớp: CÔNG NGHỆ THÔNG TIN

Khoá:61

Tp. Hồ Chí Minh, năm 2024

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: NGHIÊN CỨU VÀ ỨNG DỤNG CÔNG CỤ OR-TOOLS**

**VÀO XÂY DỰNG HỆ THỐNG ĐIỀU PHỐI NGUỒN LỰC TỰ ĐỘNG**

Giảng viên hướng dẫn: THS. TRẦN PHONG NHÃ

Sinh viên thực hiện: NGUYỄN HOÀNG PHÁT

Lớp: CÔNG NGHỆ THÔNG TIN

Khoá:61

Tp. Hồ Chí Minh, năm 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH** Độc lập – Tự do – Hạnh phúc

# NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------\*\*\*-------

**Mã sinh viên: 6151071082 Họ tên SV: Nguyễn Hoàng Phát**

**Khóa:** 61 **Lớp:** Công nghệ thông tin

1. **Tên đề tài**

Nghiên cứu và ứng dụng công cụ OR-Tools vào xây dựng hệ thống điều phối nguồn lực tự động.

1. **Mục đích, yêu cầu**

Mục tiêu chính của đề tài là phát triển một hệ thống tự động hỗ trợ điều phối và quản lý nguồn lực logistics, nhằm tối ưu hóa hiệu suất hoạt động và giảm chi phí cho doanh nghiệp. Hệ thống cần đảm bảo tính khả thi, độ tin cậy và khả năng mở rộng khi áp dụng vào thực tế. Ngoài ra, đề tài cần đáp ứng các yêu cầu về tính bảo mật và khả năng sử dụng dễ dàng, đảm bảo rằng người dùng cuối có thể tương tác với hệ thống một cách hiệu quả mà không gặp khó khăn.

1. **Nội dung và phạm vi đề tài**

Nghiên cứu chi tiết về OR-Tools, một công cụ mạnh mẽ của Google dùng để giải quyết các bài toán tối ưu hóa. Công cụ này cung cấp các thuật toán và thư viện cần thiết để xử lý các vấn đề phức tạp trong việc điều phối nguồn lực.

Phân tích các mô hình và thuật toán của OR-Tools, OR-Tools hỗ trợ nhiều mô hình và thuật toán tối ưu hóa khác nhau như Linear Programming (LP) và Constraint Programming (CP). Việc phân tích này sẽ giúp xác định mô hình và thuật toán phù hợp nhất cho hệ thống cần phát triển.

Nghiên cứu, tham khảo các ứng dụng thực tế của OR-Tools trong các ngành công nghiệp khác nhau để hiểu rõ hơn về khả năng và giới hạn của công cụ này.

Phát triển hệ thống quản lý và phân bổ nguồn lực logistics tự động. Hệ thống này cần bao gồm các chức năng như khai báo thông tin, giám sát hoạt động điều phối nguồn lực và điều phối xe cho tuyến.

1. **Công nghệ, công cụ và ngôn ngữ lập trình**

Công cụ: OR-Tools của Google Developers là công cụ chính được sử dụng để giải quyết các bài toán tối ưu hóa trong hệ thống.

Ngôn ngữ lập trình: Java sẽ được sử dụng cho backend, đảm bảo tính ổn định và khả năng mở rộng của hệ thống. Javascript sẽ được sử dụng cho giao diện website và ứng dụng di động, đảm bảo tính linh hoạt và khả năng tương thích cao.

Framework: Spring Boot sẽ hỗ trợ phát triển backend nhanh chóng và hiệu quả. ReactJS và React Native sẽ được sử dụng cho frontend, cung cấp giao diện người dùng thân thiện và linh hoạt.

Cơ sở dữ liệu: PostgreSQL sẽ quản lý dữ liệu của hệ thống, đảm bảo khả năng lưu trữ và truy xuất dữ liệu một cách hiệu quả và bảo mật.

1. **Các kết quả chính dự kiến sẽ đạt được và ứng dụng**

Hệ thống đảm bảo khả năng tự động hóa cao trong quá trình điều phối và quản lý nguồn lực bao gồm việc tự động điều chỉnh kế hoạch theo nhu cầu thực tế, giảm thiểu sự can thiệp của con người và tối ưu hóa quy trình logistics.

Tối ưu hóa quy trình logistics, giúp nâng cao hiệu suất và chất lượng dịch vụ cho doanh nghiệp. Hệ thống cần hỗ trợ việc ra quyết định dựa trên các dữ liệu thực tế và thuật toán tối ưu hóa, đảm bảo rằng nguồn lực được sử dụng một cách hiệu quả nhất.

Hỗ trợ việc ra quyết định dựa trên các dữ liệu thực tế và các thuật toán tối ưu hóa, giúp nâng cao độ chính xác và hiệu quả trong quá trình điều phối và quản lý nguồn lực.

Hệ thống sẽ được triển khai thử nghiệm tại một hoặc nhiều doanh nghiệp logistics để đánh giá tính hiệu quả và khả năng áp dụng. Việc triển khai này sẽ giúp kiểm tra và điều chỉnh hệ thống, đảm bảo rằng nó hoạt động hiệu quả trong môi trường thực tế.

1. **Giáo viên và cán bộ hướng dẫn**

Họ tên: ThS. Trần Phong Nhã

Đơn vị công tác: Bộ môn Công nghệ Thông tin

Điện thoại: Email: tpnha@utc2.edu.vn

|  |  |
| --- | --- |
| **Ngày tháng 03 năm 2024**  **Trưởng BM Công nghệ Thông tin** | **Đã giao nhiệm vụ TKTN**  **Giáo viên hướng dẫn** |
| **ThS. Trần Phong Nhã** | **ThS. Trần Phong Nhã** |

Đã nhận nhiệm vụ TKTN

Sinh viên: Nguyễn Hoàng Phát Ký tên:

Điện thoại: 0817937865 Email: nguyenphatbe@gmail.com

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn Trường Đại học Giao thông Vận tải Phân hiệu tại TP.Hồ Chí Minh, Bộ môn Công nghệ Thông tin đã tạo điều kiện thuận lợi và cung cấp một môi trường học tập, nghiên cứu chuyên nghiệp và nhiều cảm hứng. Trong suốt quá trình học tập tại trường, em đã nhận được rất nhiều sự giúp đỡ và hỗ trợ từ nhà trường, giúp em có được nền tảng kiến thức vững chắc để hoàn thành đồ án tốt nghiệp này.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc tới các thầy cô trong Bộ môn Công nghệ Thông tin. Những bài giảng, những buổi thảo luận, và những chia sẻ kinh nghiệm quý báu của các thầy cô không chỉ giúp em hiểu sâu hơn về kiến thức chuyên môn mà còn khơi dậy niềm đam mê và sáng tạo. Sự tận tâm và nhiệt huyết của các thầy cô là nguồn động viên to lớn giúp em vượt qua những khó khăn trong quá trình học tập.

Em xin bày tỏ lòng biết ơn đặc biệt đến giảng viên hướng dẫn, thầy ThS. Trần Phong Nhã. Thầy đã dành rất nhiều thời gian và tâm huyết để hướng dẫn, chỉ bảo em trong suốt quá trình thực hiện đồ án. Sự tận tình và kiên nhẫn của thầy trong việc giải đáp các thắc mắc, cũng như những góp ý và định hướng quý báu đã giúp em hoàn thiện đồ án một cách tốt nhất. Những lời khuyên và động viên của thầy không chỉ là nguồn cảm hứng mà còn là động lực to lớn giúp em nỗ lực không ngừng để hoàn thành nhiệm vụ.

Cuối cùng, em xin kính chúc Trường Đại học Giao thông Vận tải Phân hiệu tại TP.Hồ Chí Minh, Bộ môn Công nghệ Thông tin, các thầy cô và thầy ThS. Trần Phong Nhã luôn dồi dào sức khỏe, hạnh phúc và thành công trong sự nghiệp giáo dục và nghiên cứu.

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

|  |
| --- |
| ***Tp. Hồ Chí Minh, ngày ….… tháng ….… năm ….…***  **Giáo viên hướng dẫn**  **ThS. Trần Phong Nhã** |

**MỤC LỤC**

[NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP i](#_Toc169774865)

[MỞ ĐẦU 1](#_Toc169774866)

[**1.** **Tổng quan về đề tài** 1](#_Toc169774867)

[**2.** **Mục tiêu nghiên cứu** 1](#_Toc169774868)

[**3.** **Phạm vi đề tài** 3](#_Toc169774869)

[**4.** **Cấu trúc báo cáo thực tập tốt nghiệp** 4](#_Toc169774870)

[CHƯƠNG 1. TÌM HIỂU CÔNG CỤ OR-TOOLS 5](#_Toc169774871)

[**1.1** **Giới thiệu về OR-Tools** 5](#_Toc169774872)

[**1.2** **Các mô hình của OR-Tools** 5](#_Toc169774873)

[1.2.1 GLOP 5](#_Toc169774874)

[1.2.2 CP-SAT 7](#_Toc169774875)

[1.2.3 SCIP 9](#_Toc169774876)

[1.2.4 GLPK 10](#_Toc169774877)

[**1.3** **Các thuật toán trong OR-Tools** 11](#_Toc169774878)

[1.3.1 Heuristic 11](#_Toc169774879)

[1.3.2 Backtracking 13](#_Toc169774880)

[1.3.3 Propagation 15](#_Toc169774881)

[1.3.4 Branch-and-Bound 16](#_Toc169774882)

[CHƯƠNG 2. ỨNG DỤNG OR-TOOLS GIẢI QUYẾT BÀI TOÁN ASSIGNMENT PROBLEM 18](#_Toc169774883)

[**2.1** **Tổng quan bài toán Assignment** 18](#_Toc169774884)

[**2.2** **Lựa chọn mô hình cho bài toán** 19](#_Toc169774885)

[**2.3** **Bài toán tìm nhà cung cấp xe phù hợp** 21](#_Toc169774886)

[2.3.1 Phân tích bài toán 21](#_Toc169774887)

[2.3.2 Giải quyết bài toán sử dụng ngôn ngữ Java 22](#_Toc169774888)

[**2.4** **Bài toán tìm nhân viên phù hợp** 30](#_Toc169774889)

[2.4.1 Phân tích bài toán 30](#_Toc169774890)

[2.4.2 Giải quyết bài toán sử dụng ngôn ngữ Java 31](#_Toc169774891)

[CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 37](#_Toc169774892)

[**3.1** **Sơ đồ phân rã chức năng – BFD** 37](#_Toc169774893)

[**3.2** **Sơ đồ luồng dữ liệu – DFD** 39](#_Toc169774894)

[3.2.1 Mức ngữ cảnh 39](#_Toc169774895)

[3.2.2 Mức đỉnh 40](#_Toc169774896)

[3.2.3 Mức dưới đỉnh 40](#_Toc169774897)

[3.2.3.1 Điều phối tuyến 40](#_Toc169774898)

[3.2.3.2 Khai báo thông tin 42](#_Toc169774899)

[3.2.3.3 Điều phối xe 43](#_Toc169774900)

[**3.3** **Sơ đồ use-case** 44](#_Toc169774901)

[**3.4** **Mô hình ERD** 46](#_Toc169774902)

[CHƯƠNG 4. PHÁT TRIỂN ỨNG DỤNG 49](#_Toc169774903)

[**4.1** **Công cụ phát triển** 49](#_Toc169774904)

[4.1.1 Visual Studio Code 49](#_Toc169774905)

[4.1.2 DBeaver 50](#_Toc169774906)

[4.1.3 Draw.io 51](#_Toc169774907)

[**4.2** **Phát triển back-end** 52](#_Toc169774908)

[4.2.1 Cơ sở dữ liệu 52](#_Toc169774909)

[4.2.1.1 PostgreSQL 52](#_Toc169774910)

[4.2.1.2 Model hệ thống 53](#_Toc169774911)

[4.2.2 Công nghệ Spring Boot 56](#_Toc169774912)

[4.2.2.1 Repository 57](#_Toc169774913)

[4.2.2.2 Service 58](#_Toc169774914)

[4.2.2.3 Controller 59](#_Toc169774915)

[**4.3** **Phát triển front-end website** 61](#_Toc169774916)

[4.3.1 Công nghệ ReactJS 62](#_Toc169774917)

[4.3.2 Giao diện điều phối tuyến 64](#_Toc169774918)

[4.3.3 Giao diện thông tin khai báo 65](#_Toc169774919)

[4.3.4 Giao diện điều phối xe 66](#_Toc169774920)

[**4.4** **Phát triển front-end ứng dụng di động** 67](#_Toc169774921)

[4.4.1 Công nghệ React Native 67](#_Toc169774922)

[4.4.2 Quản lý thông tin xe (Loại xe và xe) 69](#_Toc169774923)

[4.4.3 Điều phối xe 70](#_Toc169774924)

[KẾT QUẢ VÀ KIẾN NGHỊ 71](#_Toc169774925)

[**1.** **Kết quả đạt được** 71](#_Toc169774926)

[**2.** **Hạn chế** 71](#_Toc169774927)

[**3.** **Kiến nghị** 71](#_Toc169774928)

[TÀI LIỆU THAM KHẢO 73](#_Toc169774929)

**DANH MỤC CHỮ VIẾT TẮT**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mô tả** | **Ý nghĩa** | **Ghi chú** |
| 1 | GLOP | Google Linear Optimization Package | Mô hình của Or-Tools |
| 2 | CP-SAT | Constraint Programming Satisfiability | Mô hình của Or-Tools |
| 3 | SCIP | Solving Constraint Integer Programs | Mô hình của Or-Tools |
| 4 | GLPK | GNU Linear Programming Kit | Mô hình tích hợp Or-Tools |
| 5 | IP | Integer Programming |  |
| 6 | CP | Constraint Programming |  |
| 7 | SAT | Satisfiability |  |
| 8 | LP | Linear Programming |  |
| 9 | CDCL | Conflict-Driven Clause Learning |  |
| 10 | MIP | Mixed-Integer Programming |  |
| 11 | CNF | Conjunctive Normal Form |  |
| 12 | BFD | Business Function Diagram |  |
| 13 | DFD | Data Flow Diagram |  |

**DANH MỤC BẢNG BIỂU**

[Bảng 2. 1 So sánh mô hình MIP và CP-SAT 21](#_Toc169774970)

**DANH MỤC SƠ ĐỒ**

[Sơ đồ 3. 1 Sơ đồ phân rã chức năng 37](#_Toc169775016)

[Sơ đồ 3. 2 Mức ngữ cảnh (Mức 0) 39](#_Toc169775017)

[Sơ đồ 3. 3 Mức đỉnh (Mức 1) 40](#_Toc169775018)

[Sơ đồ 3. 4 Mức dưới đỉnh - Điều phối tuyến 41](#_Toc169775019)

[Sơ đồ 3. 5 Mức dưới đỉnh - Khai báo thông tin 42](#_Toc169775020)

[Sơ đồ 3. 6 Mức dưới đỉnh - Điều phối xe 43](#_Toc169775021)

[Sơ đồ 3. 7 Sơ đồ use case hệ thống 44](#_Toc169775022)

[Sơ đồ 3. 8 Mô hình ERD 46](#_Toc169775023)

**DANH MỤC HÌNH ẢNH**

[Hình 1. 1 Công cụ OR-Tools 5](#_Toc169775051)

[Hình 1. 2 Solving Constraint Integer Programs 9](#_Toc169775052)

[Hình 1. 3 GNU Linear Programming Kit 10](#_Toc169775053)

[Hình 2. 1 Bài toán Assignment 19](#_Toc169775109)

[Hình 4. 1 Logo Visual Studio Code 49](#_Toc169775083)

[Hình 4. 2 Logo công cụ DBeaver 50](#_Toc169775084)

[Hình 4. 3 Logo công cụ draw.io 51](#_Toc169775085)

[Hình 4. 4 Logo PostgreSQL 52](#_Toc169775086)

[Hình 4. 5 Logo Spring Boot 56](#_Toc169775087)

[Hình 4. 6 Logo ReactJS 62](#_Toc169775088)

[Hình 4. 7 Tổ chức ReactJS của hệ thống 63](#_Toc169775089)

[Hình 4. 8 Màn hình điều phối tuyến chưa gán thông tin 64](#_Toc169775090)

[Hình 4. 9 Màn hình điều phối tuyến đã gán thông tin 65](#_Toc169775091)

[Hình 4. 10 Màn hình thông tin khai báo Bảng giá dịch vụ của các đối tác 65](#_Toc169775092)

[Hình 4. 11 Màn hình thông tin khai báo Nhân viên 66](#_Toc169775093)

[Hình 4. 12 Màn hình khai báo Công nợ nhân viên 66](#_Toc169775094)

[Hình 4. 13 Màn hình điều phối xe vào tuyến 66](#_Toc169775095)

[Hình 4. 14 Logo React Native 67](#_Toc169775096)

[Hình 4. 15 Tổ chức React Native của hệ thống 68](#_Toc169775097)

[Hình 4. 16 Màn hình quản lý loại xe và xe 69](#_Toc169775098)

[Hình 4. 17 Màn hình điều phối xe của đối tác 70](#_Toc169775099)

# MỞ ĐẦU

## **Tổng quan về đề tài**

Trong bối cảnh hiện nay, việc quản lý kết hợp điều phối nguồn lực bao gồm xe tải và nhân viên đối với các kho hàng ngày càng trở nên phức tạp. Yêu cầu về tự động hóa trong quản lý logistics và vận chuyển nhằm tối ưu hóa hiệu suất là cực kỳ quan trọng. Và một trong những điều quan trọng trong lĩnh vực này là bài toán Assignment (gán nhiệm vụ), đặt ra câu hỏi: “Làm thế nào để phân bổ một tập hợp các tài nguyên (như xe tải và nhân viên) cho một tập hợp các nhiệm vụ (như các tuyến giao hàng) sao cho tổng chi phí và thời gian là tối ưu nhất?”

Trong ngữ cảnh quản lý kho hàng và điều phối nguồn lực, bài toán này có thể được áp dụng để tự động hóa quy trình tìm các nhà cung cấp xe và nhân viên phù hợp cho các tuyến giao hàng. Thay vì phụ thuộc vào quyết định của nhân viên về cách phân bổ nguồn lực cho mỗi tuyến giao hàng, một hệ thống tự động sử dụng thuật toán tối ưu hóa có thể cải thiện quy trình này, giảm thiểu thời gian phân bổ, giảm chi phí phân bổ và tăng cường hiệu suất tổng thể của quá trình logistics.

Google OR-Tools là một công cụ tiên tiến được phát triển để giải quyết bài toán này. Việc ứng dụng OR-Tools vào bài toán Assignment là một bước đi chiến lược để tối ưu hóa hiệu suất và giảm thiểu chi phí. Thuật toán tối ưu của OR-Tools giúp tự động hóa quy trình phân bổ nguồn lực, như xe tải và nhân viên, một cách hiệu quả và chính xác hơn. Như vậy, không chỉ giảm bớt gánh nặng công việc thủ công mà còn giảm thiểu sai sót, từ đó nâng cao hiệu quả hoạt động.

Khả năng điều chỉnh nhanh chóng của OR-Tools khi có sự thay đổi đột ngột trong nhu cầu khách hàng hay điều kiện vận hành giúp duy trì dịch vụ liên tục và ổn định. Như vậy sẽ cải thiện dịch vụ khách hàng, nâng cao mức độ hài lòng và tạo lợi thế cạnh tranh. Khả năng mở rộng và tùy chỉnh của OR-Tools cũng cho phép ta dễ dàng điều chỉnh hệ thống tối ưu hóa để phù hợp với các yêu cầu kinh doanh đa dạng.

## **Mục tiêu nghiên cứu**

Mục tiêu của đề tài là nghiên cứu và ứng dụng công cụ OR-Tools vào xây dựng hệ thống tự động tìm nhà cung cấp xe và nhân viên phù hợp với các tuyến giao hàng. Hệ thống này sẽ cung cấp một giao diện thân thiện để người dùng có thể xem thông tin về các tuyến giao hàng cần phục vụ, cùng với thông tin về các nhà cung cấp và nhân viên có sẵn. Dựa trên các thông tin này, hệ thống sẽ áp dụng thuật toán tối ưu hóa để gán các nhà cung cấp xe và nhân viên vào từng tuyến giao hàng sao cho tổng chi phí và thời gian là nhỏ nhất. Mục tiêu cụ thể bao gồm:

Trước hết, nghiên cứu công cụ OR-Tools của Google Developer là bước khởi đầu quan trọng. Việc phân tích các thuật toán và khả năng của OR-Tools sẽ giúp hiểu rõ về cách mà công cụ này có thể giải quyết các bài toán tối ưu hóa trong logistics. Đánh giá khả năng áp dụng của OR-Tools sẽ giúp xác định xem liệu nó có thể thực hiện hiệu quả bài toán gán nhiệm vụ trong bối cảnh thực tế hay không.

Tiếp theo, áp dụng OR-Tools vào tự động hóa tìm nhà cung cấp và nhân viên sẽ được thực hiện bằng cách xây dựng mô hình dữ liệu chi tiết. Mô hình này sẽ quản lý thông tin về các tuyến giao hàng, nhà cung cấp xe và nhân viên, bao gồm các thông tin như thời gian, kỹ năng, bảng giá, và công nợ. Thuật toán gán nhiệm vụ sẽ được phát triển sử dụng OR-Tools, nhằm tự động hóa quy trình phân bổ nguồn lực một cách thông minh dựa trên thông tin đầu vào. Hệ thống sẽ được tích hợp với dữ liệu thực tế từ doanh nghiệp logistics, đảm bảo tính khả thi và hiệu quả của giải pháp.

Về mặt tối ưu hóa hiệu suất, hệ thống sẽ áp dụng thuật toán tối ưu hóa trong OR-Tools để đảm bảo việc gán nhà cung cấp xe và nhân viên vào các tuyến giao hàng được thực hiện một cách hiệu quả nhất. Liên tục đánh giá hiệu suất của hệ thống và tối ưu hóa các tham số sẽ giúp đạt được hiệu quả cao nhất. So sánh với phương pháp phân bổ thủ công truyền thống sẽ cho thấy rõ ràng lợi ích cụ thể của việc áp dụng hệ thống tự động này.

Đồng thời, hệ thống sẽ được thiết kế với một giao diện thân thiện và dễ sử dụng cho người dùng. Giao diện này sẽ cho phép người dùng xem kết quả một cách trực quan và tiện lợi. Việc tối ưu hóa trải nghiệm người dùng sẽ đảm bảo rằng giao diện hệ thống không chỉ trực quan, dễ hiểu mà còn thuận tiện trong quá trình sử dụng, giúp người dùng tiết kiệm thời gian và công sức.

Cuối cùng, hệ thống sẽ cung cấp khả năng tích hợp linh hoạt với các hệ thống quản lý kho hàng và vận chuyển hiện có của doanh nghiệp. Vấn đề này sẽ được thực hiện bằng cách hỗ trợ các giao thức kết nối phổ biến, đảm bảo quá trình tích hợp diễn ra thuận lợi. Hệ thống cũng sẽ được thiết kế để dễ dàng mở rộng và điều chỉnh khi có thay đổi trong nhu cầu kinh doanh hoặc môi trường vận hành.

## **Phạm vi đề tài**

Đề tài này tập trung vào tự động hóa quy trình điều phối nguồn lực vào các tuyến giao hàng trong lĩnh vực logistics và vận chuyển. Phạm vi nghiên cứu bao gồm phát triển một hệ thống tự động có khả năng quản lý thông tin về các tuyến giao hàng, nhà cung cấp, bảng giá, nhân viên và thời gian làm việc. Hệ thống sẽ áp dụng thuật toán tối ưu hóa để gán nhà cung cấp và nhân viên phù hợp cho từng tuyến giao hàng, nhằm tối thiểu hóa tổng chi phí và thời gian.

Đối tượng nghiên cứu của đề tài là các doanh nghiệp hoạt động trong lĩnh vực logistics và vận tải, đặc biệt là những doanh nghiệp có nhu cầu tự động hóa quy trình điều phối nguồn lực và tối ưu hóa chi phí logistics bao gồm các công ty vận chuyển hàng hóa, các nhà cung cấp dịch vụ logistics bên thứ ba và các doanh nghiệp bán lẻ có nhu cầu vận chuyển hàng hóa hiệu quả. Ngoài ra còn là những người quản lý và đưa ra quyết định trong các doanh nghiệp logistics, những người đang tìm kiếm giải pháp để tối ưu hóa quy trình gán xe và nâng cao hiệu suất trong hoạt động vận chuyển và điều phối hàng hóa.

Phạm vi đề tài bao trùm cả nghiên cứu lý thuyết và ứng dụng thực tiễn của OR-Tools trong việc tối ưu hóa quy trình điều phối nguồn lực trong logistics. Hệ thống tự động được phát triển sẽ mang lại giá trị lớn cho các doanh nghiệp logistics, giúp họ cải thiện hiệu suất hoạt động, giảm chi phí và nâng cao chất lượng dịch vụ. Tập trung vào đối tượng là các doanh nghiệp và người quản lý trong ngành logistics đảm bảo rằng các giải pháp được đề xuất sẽ đáp ứng đúng nhu cầu và thách thức thực tế của ngành.

## **Cấu trúc báo cáo thực tập tốt nghiệp**

* 1. Mở đầu
  2. Chương 1: Tìm hiểu công cụ or-tools
  3. Chương 2: Ứng dụng or-tools giải quyết bài toán assignment problem
  4. Chương 3: Phân tích thiết kế hệ thống
  5. Chương 4: Phát triển ứng dụng
  6. Kết quả và kiến nghị
  7. Tài liệu tham khảo

# CHƯƠNG 1. TÌM HIỂU CÔNG CỤ OR-TOOLS

## **Giới thiệu về OR-Tools**



Hình 1. 1 Công cụ OR-Tools

OR-Tools là một bộ công cụ tối ưu hóa mạnh mẽ và mã nguồn mở được Google phát triển, nhằm giúp giải quyết các bài toán phức tạp trong nhiều lĩnh vực, đặc biệt là trong lĩnh vực logistics và vận tải. Với sự tích hợp của nhiều thuật toán tối ưu hóa, OR-Tools mang lại cho người dùng một công cụ linh hoạt và hiệu quả để tìm ra các giải pháp tối ưu cho các vấn đề đa dạng, từ quản lý nguồn lực đến lập lịch công việc.

Trong lĩnh vực logistics và vận tải, OR-Tools đặc biệt nổi bật với khả năng giải quyết các bài toán như tối ưu hóa lộ trình vận chuyển (VRP) và bài toán gán nhiệm vụ (Assignment Problem). Công cụ này không chỉ giúp tối ưu hóa việc phân chia nguồn lực một cách hiệu quả, giảm thiểu chi phí và tăng cường hiệu suất hoạt động, mà còn cung cấp khả năng linh hoạt và tích hợp dễ dàng vào các hệ thống tồn tại.

Với OR-Tools, người dùng có thể tận dụng sức mạnh của các thuật toán tối ưu hóa tiên tiến để tìm ra các giải pháp tối ưu cho các thách thức phức tạp trong quản lý nguồn lực và tổ chức vận tả, giúp nâng cao hiệu suất, giảm thiểu chi phí và tạo ra những lợi ích rõ ràng cho các doanh nghiệp hoạt động trong lĩnh vực này. Với sự linh hoạt và khả năng tích hợp, OR-Tools không chỉ là một công cụ, mà còn là một đối tác đáng tin cậy trong việc tối ưu hóa quy trình và tăng cường hiệu suất trong lĩnh vực logistics và vận tải. [1]

## **Các mô hình của OR-Tools**

### GLOP

GLOP là một bộ giải toán tối ưu tuyến tính được phát triển bởi Google trong công cụ OR-Tools. GLOP được thiết kế để giải các bài toán quy hoạch tuyến tính, một loại bài toán tối ưu hóa trong đó hàm mục tiêu và các ràng buộc đều là các hàm tuyến tính.

Các thành phần chính của GLOP bao gồm hàm mục tiêu, các ràng buộc và các biến. Hàm mục tiêu là hàm tuyến tính cần được tối ưu hóa, có thể là tối đa hóa hoặc tối thiểu hóa. Ví dụ, một hàm mục tiêu có thể là c­1\*x1 + c2\*x2 + ... + cn\*xn, trong đó c1, c2, ..., cn là các hệ số và x1, x2, ..., xn là các biến. Các ràng buộc là những điều kiện tuyến tính mà các biến phải tuân thủ, có thể ở dạng bất đẳng thức hoặc đẳng thức, như a1\*x1 + a2\*x2 + ... + an\*xn ≤ b. Các biến là các biến số trong bài toán tối ưu hóa, có thể có giới hạn trên và giới hạn dưới.

GLOP nổi bật với các tính năng như hiệu suất cao, độ chính xác và dễ sử dụng. Bộ giải này được tối ưu hóa để có thể xử lý các bài toán quy hoạch tuyến tính lớn với hiệu suất cao, sử dụng các kỹ thuật số học tiên tiến để đảm bảo độ chính xác của kết quả. Tích hợp trong OR-Tools, GLOP có API dễ sử dụng và hỗ trợ nhiều ngôn ngữ lập trình như Python, C++ và Java, giúp người dùng dễ dàng triển khai và sử dụng trong các ứng dụng thực tế.

Dưới đây là một ví dụ cơ bản về cách sử dụng GLOP. Đoạn mã tạo ra một bộ giải GLOP, định nghĩa các biến, thiết lập hàm mục tiêu và các ràng buộc, sau đó giải bài toán và in kết quả:

solver = pywraplp.Solver.CreateSolver('GLOP')

# Tạo các biến với giới hạn

x = solver.NumVar(0, infinity, 'x')

y = solver.NumVar(0, infinity, 'y')

# Định nghĩa hàm mục tiêu: tối đa hóa 3 \* x + 4 \* y

solver.Maximize(3 \* x + 4 \* y)

# Thêm các ràng buộc

solver.Add(x + 2 \* y <= 14)

solver.Add(3 \* x - y >= 0)

solver.Add(x - y <= 2)

# Giải bài toán

status = solver.Solve()

# Kiểm tra và in kết quả

if status == pywraplp.Solver.OPTIMAL:

print('Giá trị tối ưu:')

print('x =', x.solution\_value())

print('y =', y.solution\_value())

print('Giá trị hàm mục tiêu =', solver.Objective().Value())

else:

print('Không tìm thấy giải pháp tối ưu.')

Trong quản lý chuỗi cung ứng, GLOP giúp tối ưu hóa việc quản lý kho, vận chuyển và sản xuất, đảm bảo số lượng sản phẩm cần sản xuất và vận chuyển để tối thiểu hóa chi phí và đáp ứng nhu cầu khách hàng. Lập kế hoạch sản xuất cũng là một ứng dụng quan trọng, nơi GLOP tối ưu hóa lịch trình sản xuất nhằm giảm thiểu chi phí nguyên liệu, nhân công và thời gian sản xuất.

Trong quản lý vận tải, GLOP giải quyết các bài toán tối ưu hóa tuyến đường và phân bổ phương tiện, giúp giảm chi phí vận tải và cải thiện hiệu quả logistics. Quản lý nguồn lực trong tổ chức và doanh nghiệp cũng được tối ưu hóa nhờ GLOP, đảm bảo rằng nhân lực, tài chính và vật liệu được sử dụng hiệu quả nhất để đạt được mục tiêu. Trong lĩnh vực tài chính, GLOP hỗ trợ tối ưu hóa danh mục đầu tư, quản lý rủi ro và lập kế hoạch tài chính, giúp các nhà quản lý quỹ xác định tỷ lệ đầu tư tối ưu vào các loại tài sản khác nhau.

Ngoài ra, GLOP còn được ứng dụng trong lập lịch công việc, tạo ra các lịch làm việc tối ưu cho nhân viên để đảm bảo phân bố công bằng và hiệu quả, đặc biệt hữu ích trong các ngành như bán lẻ, y tế và dịch vụ khách hàng. [2]

### CP-SAT

CP-SAT là một trong những mô hình mạnh mẽ và linh hoạt nhất trong OR-Tools của Google. CP-SAT kết hợp giữa lập trình ràng buộc (CP) và khả năng giải bài toán thỏa mãn (SAT), tạo ra một công cụ hiệu quả cho việc giải các bài toán tối ưu hóa tổ hợp và các vấn đề về khả năng thỏa mãn ràng buộc. Sự kết hợp này giúp CP-SAT không chỉ mạnh mẽ trong xử lý các bài toán có quy mô lớn mà còn đảm bảo tính ổn định và hiệu quả cao trong quá trình giải quyết bài toán.

CP-SAT nổi bật với ba đặc điểm chính: phương pháp tiên tiến, hiệu quả cao và tính ổn định. Đầu tiên, CP-SAT sử dụng sự kết hợp giữa lập trình ràng buộc và khả năng giải bài toán thỏa mãn, giúp cải thiện tốc độ hội tụ và hiệu quả giải quyết bài toán. Lập trình ràng buộc cho phép mô hình hóa các bài toán thông qua việc xác định các biến, miền giá trị của biến và các ràng buộc giữa các biến. Công thức cơ bản cho một bài toán CP là xác định biến xi∈Di và thỏa mãn các ràng buộc Cj(x1,x2,…,xn). SAT solver tích hợp trong CP-SAT sử dụng các thuật toán mạnh mẽ, như CDCL, để tìm ra các giải pháp thỏa mãn tất cả các ràng buộc một cách nhanh chóng và hiệu quả, đặc biệt trong việc giải các bài toán tối ưu hóa tổ hợp và logic phức tạp. Thứ hai, CP-SAT được thiết kế để tối ưu hóa việc sử dụng bộ nhớ và thời gian xử lý, giúp giải quyết nhanh chóng các bài toán có nhiều biến và ràng buộc, nó đặc biệt quan trọng trong các ứng dụng thực tế đòi hỏi xử lý lượng dữ liệu lớn. Cuối cùng, tính ổn định của CP-SAT được đảm bảo thông qua các kỹ thuật primal-dual và gradient, giúp tránh các vấn đề về hội tụ mà một số phương pháp gradient có thể gặp phải. Các đặc điểm này làm cho CP-SAT trở thành một công cụ mạnh mẽ và linh hoạt trong việc giải quyết các bài toán tối ưu hóa phức tạp.

Một ví dụ cụ thể về cách sử dụng CP-SAT trong OR-Tools là bài toán tối ưu hóa lịch làm việc. Trong bài toán này, CP-SAT có thể được sử dụng để tạo ra một lịch làm việc tối ưu cho một nhóm nhân viên, đảm bảo rằng mỗi nhân viên chỉ làm một ca mỗi ngày và mỗi ca làm việc đều được đảm nhận bởi một nhân viên. Bằng cách khởi tạo các biến đại diện cho các ca làm việc, thiết lập các ràng buộc về số lượng ca làm việc mỗi ngày và giải bài toán bằng bộ giải CP-SAT, ta có thể tìm ra lịch làm việc tối ưu một cách hiệu quả.

CP-SAT có nhiều ứng dụng quan trọng trong các lĩnh vực khác nhau nhờ khả năng xử lý hiệu quả các bài toán tối ưu hóa tổ hợp và các vấn đề về khả năng thỏa mãn ràng buộc. Trong lập lịch và quản lý nhân sự, CP-SAT giúp tối ưu hóa lịch làm việc của nhân viên, đảm bảo các ca làm việc không chồng chéo và đáp ứng yêu cầu về thời gian làm việc và nghỉ ngơi. Trong lĩnh vực sản xuất, CP-SAT tối ưu hóa quy trình sản xuất, sử dụng hiệu quả nguyên vật liệu và thời gian, từ đó giảm chi phí và tăng năng suất. Ngoài ra, trong quản lý chuỗi cung ứng, CP-SAT hỗ trợ tối ưu hóa quá trình vận chuyển và lưu kho, đảm bảo hàng hóa được phân phối đúng thời gian và địa điểm với chi phí thấp nhất. Trong thiết kế mạch điện và hệ thống VLSI, CP-SAT giúp tối ưu hóa bố trí mạch và giảm thiểu sự can thiệp giữa các thành phần điện tử. [3]

### SCIP



Hình 1. 2 Solving Constraint Integer Programs

SCIP được thiết kế để giải các bài toán lập trình số nguyên (IP) và lập trình hỗn hợp số nguyên (MIP). SCIP là một trong những bộ giải tối ưu hóa tiên tiến nhất hiện nay, nổi bật với khả năng xử lý các bài toán có độ phức tạp cao và quy mô lớn.

SCIP hỗ trợ cả lập trình số nguyên và lập trình hỗn hợp số nguyên, cho phép giải quyết các bài toán có biến nguyên và biến liên tục cùng tồn tại, giúp SCIP trở thành công cụ lý tưởng cho nhiều loại bài toán tối ưu hóa thực tế. Công thức cơ bản cho bài toán MIP có dạng: tối thiểu hóa cTx với các ràng buộc Ax ≤ b và các biến xi​ thuộc tập số nguyên. SCIP sử dụng nhiều phương pháp tối ưu như cắt mặt phẳng (cutting planes), phân nhánh và cận (branch-and-bound), và heuristics, kết hợp chúng để tìm ra giải pháp tối ưu một cách hiệu quả.

Một đặc điểm nổi bật khác của SCIP là tính linh hoạt và tùy biến cao. SCIP cho phép người dùng tùy chỉnh và mở rộng thông qua thêm các ràng buộc, biến và các phương pháp tối ưu riêng, mang lại sự linh hoạt trong giải quyết các bài toán đa dạng và phức tạp. SCIP cũng được tối ưu hóa để xử lý các bài toán lớn với hàng nghìn biến và ràng buộc, sử dụng hiệu quả bộ nhớ và thời gian xử lý, giúp giải quyết nhanh chóng các bài toán có quy mô lớn.

SCIP có nhiều ứng dụng quan trọng trong các lĩnh vực khác nhau nhờ khả năng xử lý hiệu quả các bài toán tối ưu hóa tổ hợp và các vấn đề về khả năng thỏa mãn ràng buộc. Trong lập lịch và quản lý nhân sự, SCIP giúp tối ưu hóa lịch làm việc của nhân viên, đảm bảo các ca làm việc không chồng chéo và đáp ứng yêu cầu về thời gian làm việc và nghỉ ngơi. Trong lĩnh vực sản xuất, SCIP tối ưu hóa việc lập kế hoạch sản xuất, quản lý nguyên vật liệu và thời gian sản xuất để đảm bảo hiệu quả và giảm chi phí. Trong quản lý chuỗi cung ứng, SCIP hỗ trợ tối ưu hóa quá trình vận chuyển và lưu kho, đảm bảo hàng hóa được phân phối đúng thời gian và địa điểm với chi phí thấp nhất. SCIP cũng được ứng dụng trong thiết kế mạng lưới giao thông, hệ thống viễn thông và mạng máy tính, đảm bảo hiệu quả trong việc sử dụng tài nguyên và đáp ứng nhu cầu.

Một ví dụ đơn giản về cách sử dụng SCIP trong OR-Tools để giải một bài toán lập trình số nguyên. Giả sử ta muốn tối ưu hóa hàm mục tiêu 3x+4y với các ràng buộc x+2y ≤ 14, 3x−y ≥ 0, và x−y ≤ 2. Bài toán có thể được giải bằng cách khởi tạo các biến nguyên x và y, thiết lập các ràng buộc, và gọi bộ giải SCIP để tìm giải pháp tối ưu. Kết quả sẽ cho thấy giá trị tối ưu của x và y sao cho hàm mục tiêu đạt giá trị cao nhất.

SCIP mang lại nhiều lợi ích quan trọng như khả năng xử lý các bài toán có cấu trúc phức tạp và nhiều ràng buộc, sử dụng hiệu quả các phương pháp tối ưu hóa tiên tiến để tìm ra giải pháp tối ưu nhanh chóng, giúp SCIP trở thành một công cụ mạnh mẽ và linh hoạt, phù hợp với nhiều ứng dụng thực tế đòi hỏi tối ưu hóa và thỏa mãn ràng buộc phức tạp, từ quản lý chuỗi cung ứng, lập kế hoạch sản xuất, đến quản lý tài sản và thiết kế mạng lưới. Sự tích hợp và tùy biến cao của SCIP giúp các doanh nghiệp và tổ chức đạt được hiệu quả cao nhất trong các hoạt động của mình, tối ưu hóa nguồn lực và nâng cao khả năng cạnh tranh trên thị trường. [4]

### GLPK



Hình 1. 3 GNU Linear Programming Kit

GLPK chuyên dụng cho giải các bài toán quy hoạch tuyến tính (LP) và quy hoạch hỗn hợp số nguyên (MIP). GLPK là một phần mềm mã nguồn mở, nổi tiếng với tính dễ sử dụng và khả năng xử lý các bài toán tối ưu hóa có quy mô vừa và nhỏ. Được tích hợp trong OR-Tools, GLPK cung cấp một giải pháp tối ưu hóa hiệu quả và linh hoạt cho nhiều loại bài toán.

GLPK hỗ trợ giải các bài toán quy hoạch tuyến tính (LP) và quy hoạch hỗn hợp số nguyên (MIP), cho phép giải quyết các bài toán có cả biến liên tục và biến nguyên. Công thức cơ bản của một bài toán LP là tối thiểu hóa cTx với các ràng buộc Ax ≤ b, và x ≥ 0, trong khi bài toán MIP bổ sung thêm ràng buộc rằng một số hoặc tất cả các biến xi​ phải là số nguyên. Khả năng này giúp GLPK trở thành công cụ lý tưởng cho nhiều loại bài toán tối ưu hóa thực tế.

GLPK được tích hợp dễ dàng trong OR-Tools, cung cấp một giao diện đơn giản để định nghĩa và giải các bài toán tối ưu hóa giúp người dùng dễ dàng triển khai và thử nghiệm các mô hình tối ưu hóa khác nhau. Mặc dù không phải là công cụ mạnh mẽ nhất cho các bài toán có quy mô rất lớn, GLPK lại hoạt động rất hiệu quả cho các bài toán có quy mô vừa và nhỏ nên GLPK trở thành một lựa chọn lý tưởng cho các ứng dụng không đòi hỏi khối lượng tính toán quá lớn.

GLPK có nhiều ứng dụng quan trọng trong các lĩnh vực khác nhau nhờ khả năng xử lý hiệu quả các bài toán tối ưu hóa tổ hợp và các vấn đề về khả năng thỏa mãn ràng buộc. Trong lập kế hoạch sản xuất, GLPK giúp tối ưu hóa quy trình sản xuất, từ lập kế hoạch sản xuất đến quản lý nguyên vật liệu và thời gian sản xuất, nhằm giảm thiểu chi phí và nâng cao hiệu suất. Trong quản lý chuỗi cung ứng, GLPK được sử dụng để tối ưu hóa các vấn đề liên quan đến vận chuyển, lưu kho và phân phối sản phẩm, đảm bảo hàng hóa được vận chuyển đúng thời gian và địa điểm với chi phí thấp nhất. GLPK cũng hỗ trợ lập kế hoạch và quản lý dự án, từ việc phân bổ nguồn lực đến lập lịch công việc, giúp đảm bảo dự án hoàn thành đúng thời hạn và trong ngân sách. Trong thiết kế mạng lưới, GLPK có thể tối ưu hóa các yếu tố như vị trí đặt trạm, đường truyền và phân bổ băng thông, nhằm đảm bảo hiệu quả và tiết kiệm chi phí. [5]

## **Các thuật toán trong OR-Tools**

Thuật toán được sử dụng trong các mô hình OR-Tools của Google rất nhiều, ở phần này chỉ tập trung vào các thuật toán có liên quan đến đề tài.

### Heuristic

Heuristic là những kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, học hỏi hoặc khám phá. Chúng được thiết kế để đưa ra các giải pháp mà không đảm bảo chúng là tối ưu. Khi nghiên cứu không có tính thực tế, các phương pháp heuristic giúp tăng tốc quá trình tìm kiếm giải pháp hợp lý thông qua việc giảm bớt sự nhận thức vấn đề khi đưa ra quyết định. Ví dụ về phương pháp này bao gồm sử dụng các quy tắc ngón tay cái, giả thuyết, phán đoán trực giác, khuôn mẫu hoặc nhận thức thông thường.

Thuật giải Heuristic là một sự mở rộng của khái niệm thuật toán, mang các đặc điểm đáng chú ý. Thứ nhất, thuật giải này thường tìm được lời giải tốt nhưng không đảm bảo là lời giải tốt nhất. Thứ hai, giải bài toán bằng thuật giải Heuristic thường dễ dàng và nhanh chóng hơn so với các giải thuật tối ưu, do đó chi phí cũng thấp hơn. Thứ ba, thuật giải Heuristic thường thể hiện cách suy nghĩ và hành động tự nhiên, gần gũi với con người.

Có nhiều phương pháp để xây dựng một thuật giải Heuristic, dựa trên một số nguyên lý cơ bản. Nguyên lý vét cạn thông minh là một trong số đó, trong một bài toán tìm kiếm có không gian tìm kiếm lớn, nguyên lý này giúp giới hạn không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu. Nguyên lý tham lam (Greedy) là một nguyên lý khác, sử dụng tiêu chuẩn tối ưu trên phạm vi toàn cục của bài toán làm tiêu chuẩn chọn lựa hành động cho từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

Nguyên lý thứ tự cũng đóng vai trò quan trọng trong thuật giải Heuristic. Nguyên lý này thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát, nhằm nhanh chóng đạt được một lời giải tốt. Cuối cùng, hàm Heuristic là công cụ thường được sử dụng trong việc xây dựng các thuật giải Heuristic. Đây là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, người ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

Trong OR-Tools, cụ thể là CP-SAT, các heuristic được sử dụng để tăng cường hiệu quả của thuật toán quay lui khi giải quyết các bài toán ràng buộc (CSPs). Các heuristic trong OR-Tools giúp chọn biến và giá trị sao cho tìm kiếm giải pháp trở nên nhanh chóng và hiệu quả hơn.

*Heuristic chọn biến* *(Variable Selection Heuristic)* giúp chọn biến nào sẽ được gán giá trị tiếp theo bao gồm:

VSIDS (Variable State Independent Decaying Sum): Chọn các biến có tần suất xuất hiện cao nhất trong các xung đột gần đây.

Jeroslow-Wang Heuristic: Chọn các biến tối ưu hóa khả năng làm đúng các mệnh đề.

*Heuristic chọn giá trị (Value Selection Heuristic)* giúp chọn giá trị nào sẽ được gán cho biến đã chọn bao gồm:

Least Constraining Value: Chọn giá trị sao cho nó hạn chế ít nhất các lựa chọn cho các biến chưa được gán giá trị khác giúp giữ cho các biến khác có nhiều lựa chọn hơn.

Most Promising Value: Chọn giá trị có khả năng cao nhất để dẫn đến một giải pháp hợp lệ. Giá trị này thường được xác định dựa trên các thông tin hoặc kinh nghiệm trước đó. [6]

### Backtracking

Backtracking (Quay lui) là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy. Người đầu tiên đề ra thuật ngữ này (backtrack) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.

Quay lui dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng từng phần tử. Mỗi phần tử lại được chọn bằng cách thử tất cả các khả năng. Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu (Depth-First Search). [7]

Ý tưởng về quay lui như sau:

Backtracking(k) {

for([Mỗi phương án chọn i(thuộc tập D)]) {

if ([Chấp nhận i]) {

[Chọn i cho X[k]];

if ([Thành công]) {

[Đưa ra kết quả];

} else {

Backtracking(k+1);

[Bỏ chọn i cho X[k]];

}

}

}

}

Thuật toán quay lui là một kỹ thuật quan trọng trong giải quyết các bài toán ràng buộc, và nó được áp dụng trong OR-Tools, cụ thể là CP-SAT. CP-SAT giải quyết các CSPs bằng cách sử dụng thuật toán quay lui để tìm ra giải pháp tối ưu hoặc một giải pháp thỏa mãn tất cả các ràng buộc. Quy trình này có thể được mô tả như sau:

Bắt đầu từ một giải pháp rỗng hoặc một giải pháp ban đầu, thuật toán quay lui bắt đầu bằng cách chọn một biến để gán giá trị. Đối với biến đã chọn, chọn một giá trị từ miền giá trị của biến đó. Kiểm tra xem việc gán giá trị cho biến đã chọn có làm cho các ràng buộc bị vi phạm hay không. Nếu gán giá trị mới làm cho các ràng buộc bị vi phạm, quay lui đến bước trước và chọn một giá trị khác cho biến đó. Quá trình trên được lặp lại cho đến khi tìm được một giải pháp thỏa mãn tất cả các ràng buộc hoặc không còn giá trị nào để thử.

Để tăng hiệu quả của kỹ thuật quay lui, nhiều heuristics và kỹ thuật cải tiến đã được phát triển:

a. Variable Selection Heuristic (Heuristics chọn biến và giá trị)

b. Conflict-Driven Clause Learning (CDCL)

Khi gặp phải mâu thuẫn, không chỉ quay lui mà còn học từ mâu thuẫn đó bằng cách thêm các điều khoản mới (clauses) để ngăn chặn việc lặp lại mâu thuẫn tương tự. Giúp giảm thiểu số lượng lần quay lui và tăng hiệu quả của quá trình tìm kiếm.

c. Non-Chronological Backtracking (Quay Lui Không Theo Thứ Tự Thời Gian)

Thay vì quay lui đến biến gần nhất, quay lui đến biến có liên quan trực tiếp đến mâu thuẫn, giúp bỏ qua các trạng thái trung gian không cần thiết.

### Propagation

Kỹ thuật lan truyền (propagation) là một thành phần quan trọng trong các giải pháp SAT. Trong bối cảnh của SAT solving, lan truyền được sử dụng để đơn giản hóa bài toán và thu hẹp không gian tìm kiếm. Dưới đây là một mô tả chi tiết về kỹ thuật lan truyền trong SAT.

Lan truyền trong SAT là quá trình suy diễn các giá trị của biến từ các ràng buộc hiện có để đơn giản hóa bài toán. Kỹ thuật này tận dụng các thông tin có được từ các ràng buộc đã cho để xác định giá trị của các biến khác nhằm giảm số lượng các phép gán cần kiểm tra.

Có hai loại lan truyền trong SAT bao gồm:

*Unit Propagation (Lan Truyền Đơn Vị)*: Nếu một mệnh đề trong công thức CNF chỉ còn một biến chưa được gán giá trị và các biến khác đều đã được gán giá trị sao cho mệnh đề chỉ còn đúng nếu biến này được gán giá trị cụ thể, thì biến này sẽ được gán giá trị đó.

Ví dụ: Với mệnh đề (¬A ∨ B ∨ C), nếu A là true và B là false, thì để mệnh đề đúng, C phải là true. Do đó, C sẽ được gán giá trị true.

*Pure Literal Elimination (Loại Bỏ Biến Thuần Khiết)*: Nếu một biến chỉ xuất hiện ở dạng khẳng định hoặc phủ định trong toàn bộ công thức, biến này có thể được gán giá trị tương ứng mà không ảnh hưởng đến tính khả thi của bài toán.

Ví dụ: Nếu biến D chỉ xuất hiện ở dạng D (không có ¬D), ta có thể gán D = true mà không làm mất tính khả thi của công thức.

Quy trình lan truyền được định nghĩa như sau. Bắt đầu với một công thức CNF và tập các phép gán ban đầu (có thể là rỗng). Xác định các mệnh đề đơn vị (những mệnh đề chỉ còn một biến chưa được gán giá trị). Gán giá trị cần thiết cho biến này để mệnh đề trở thành true. Cập nhật công thức CNF bằng cách loại bỏ các mệnh đề đã trở thành true và đơn giản hóa các mệnh đề chứa biến này. Nếu một mệnh đề trở thành false (tất cả các biến của nó đều có giá trị làm cho nó false), bài toán không có lời giải với phép gán hiện tại. Nếu không còn mệnh đề nào để đơn giản hóa, tiếp tục với các bước giải quyết khác hoặc quay lui (backtrack) nếu cần. [8]

### Branch-and-Bound

Thuật toán branch-and-bound (nhánh và cận) là một phương pháp tối ưu hóa tổng quát được sử dụng để giải quyết các bài toán tối ưu hóa tổ hợp, đặc biệt là các bài toán NP-khó. Đây là một trong những phương pháp chủ yếu để giải quyết các bài toán như bài toán người bán hàng (Traveling Salesman Problem), bài toán lập lịch (Scheduling Problem), và nhiều bài toán khác. Branch-and-bound chính là một phương pháp cải tiến từ phương pháp Quay lui, được sử dụng để tìm nghiệm của bài toán tối ưu. Nguyên lý cơ bản của branch-and-bound dựa trên chiến lược phân chia và chinh phục (divide-and-conquer) kết hợp với việc cắt tỉa (pruning) để giảm không gian tìm kiếm. Bằng cách này, thuật toán phân chia bài toán lớn thành nhiều bài toán con nhỏ hơn, tính toán các giới hạn trên và giới hạn dưới của lời giải của các bài toán con, và loại bỏ các nhánh không thể dẫn đến lời giải tốt hơn.

Bước đầu tiên của phương pháp vẫn giống với ý tưởng của quay lui: Tìm cách biểu diễn nghiệm của bài toán dưới dạng một vector (x1,x2,…,xn), mỗi thành phần xi​ được chọn ra từ tập các Si​.

Bước tiếp theo, nếu ở phương pháp Quay lui, chỉ cần tuần tự chọn các ứng cử viên cho từng thành phần của vector nghiệm, thì ở phương pháp này, mỗi nghiệm X=(x1,x2,…,xn) của bài toán sẽ được đánh giá bằng một hàm f(X). Vì đây là bài toán tối ưu, nên mục tiêu là đi tìm nghiệm có hàm f(X) tốt nhất, thường là lớn nhất hoặc nhỏ nhất.

Giả sử, sau khi xây dựng được i thành phần của nghiệm là (x1,x2,…,xn) và chuẩn bị mở rộng nghiệm thành (x1​,x2​,…,xi​,xi+1​). Nếu như đánh giá được độ tốt của toàn bộ các nghiệm mở rộng nhánh này là (x1​,x2​,…,xi​,xi+1​,…) và biết rằng không có nghiệm nào trong nhánh này “tốt hơn” nghiệm tốt nhất tại thời điểm đó, thì việc mở rộng tiếp từ (x1,x2,…,xn) sẽ không cần thiết nữa, mà thay vào đó sẽ chuyển qua chọn tập tiếp theo cho thành phần xi​.

Bằng phương pháp trên, ta sẽ loại bỏ được những nhánh không cần thiết để không duyệt vào các phương án đó, từ đó việc tìm ra nghiệm tối ưu sẽ nhanh hơn. Tuy nhiên, việc đánh giá được “độ tốt” của các nghiệm mở rộng không phải việc đơn giản, nhưng nếu làm được như vậy thì giải thuật sẽ thực thi nhanh hơn so với quay lui. [9]

Thuật toán branch-and-bound là nền tảng cơ bản của SCIP, nhưng nó được bổ sung và mở rộng với nhiều kỹ thuật tiên tiến để cải thiện hiệu quả và khả năng giải quyết các bài toán phức tạp. Trong bước phân nhánh, SCIP chia không gian tìm kiếm theo các giá trị của các biến nguyên. Ví dụ, nếu một biến nguyên x chưa có giá trị nguyên trong nghiệm hiện tại, SCIP có thể tạo ra hai nhánh: một nhánh với x ≤ ⌊x⌋ và một nhánh khác với x ≥ ⌈x⌉. Ở bước ràng buộc, SCIP sử dụng các giải pháp thư giãn (relaxation) của bài toán gốc để tính toán các giới hạn. Thường thì bài toán tuyến tính liên tục (LP) là một dạng thư giãn của bài toán MIP. Giới hạn này cung cấp một ước lượng cho giá trị tốt nhất có thể đạt được trong nhánh đó. Quá trình cắt tỉa trong SCIP rất quan trọng để giảm không gian tìm kiếm. Nếu giới hạn dưới của một nhánh lớn hơn giới hạn trên của nghiệm tốt nhất hiện tại, nhánh đó sẽ bị cắt đi vì không thể chứa nghiệm tốt hơn.

# CHƯƠNG 2. ỨNG DỤNG OR-TOOLS GIẢI QUYẾT BÀI TOÁN ASSIGNMENT PROBLEM

## **Tổng quan bài toán Assignment**

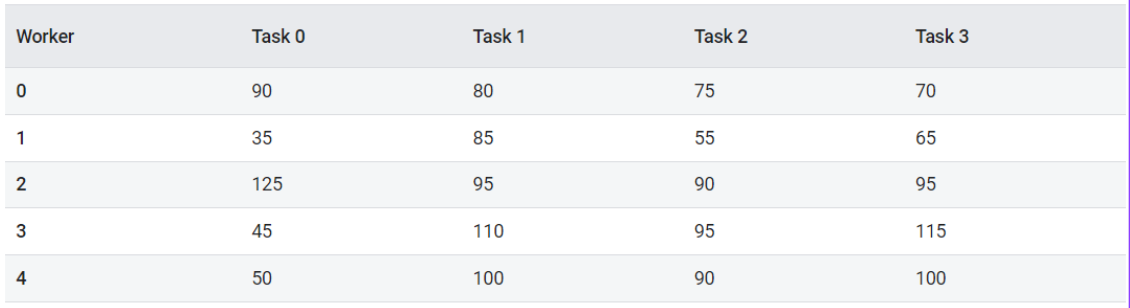
Bài toán Assignment là một bài toán quan trọng trong lĩnh vực tối ưu hóa, được áp dụng rộng rãi trong nhiều ngành công nghiệp và lĩnh vực khác nhau. Trong bối cảnh của đề tài, nó trở thành một công cụ mạnh mẽ để tự động hóa quy trình tìm nhà cung cấp xe và nhân viên vào các tuyến giao hàng trong ngành logistics và vận chuyển.

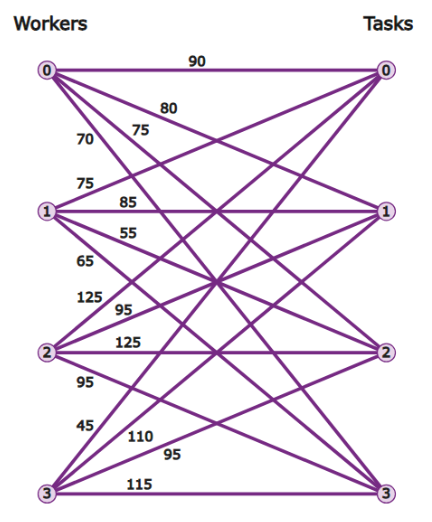
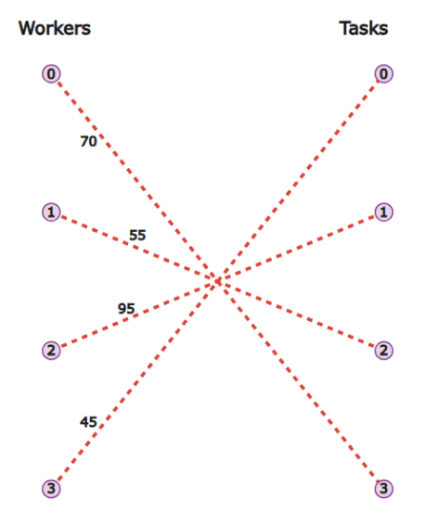
Đối với các doanh nghiệp logistics, việc điều phối các tuyến giao hàng một cách hiệu quả là một thách thức lớn. Mỗi tuyến giao hàng có thể có các yêu cầu đặc biệt về thời gian, tải trọng, hoặc các ràng buộc khác. Đồng thời, các nhà cung cấp có sẵn cũng có các hạn chế về khả năng vận chuyển, nguồn lực, và tải trọng. Do đó, việc phân bổ vào các tuyến giao hàng một cách tối ưu không chỉ giúp giảm thiểu chi phí vận chuyển mà còn đảm bảo thời gian và chất lượng dịch vụ.

Bằng cách áp dụng bài toán Assignment, các doanh nghiệp logistics có thể tự động điều phối các tuyến giao hàng sao cho mục tiêu tối ưu nhất được đạt được. Mục tiêu này có thể là giảm thiểu chi phí tổng cộng của quá trình vận chuyển, tối ưu hóa thời gian giao hàng, hoặc cân đối giữa các yếu tố khác nhau như chi phí và thời gian, giúp doanh nghiệp tăng cường hiệu suất, giảm thiểu lãng phí và tối ưu hóa lợi nhuận. [10]

Ngoài ra, tự động hóa quy trình tìm nhà cung cấp và nhân viên cũng mang lại sự linh hoạt và đáp ứng nhanh chóng hơn đối với biến động trong nhu cầu vận chuyển. Thay vì phải dành nhiều thời gian và công sức để thực hiện các quyết định điều phối thủ công, hệ thống tự động có thể đáp ứng tức thì và điều chỉnh quy trình dựa trên thông tin mới và yêu cầu thay đổi từ khách hàng.

Ví dụ: Có 5 worker và 4 tasks, với mỗi tasks có chi phí của từng worker. Vậy bài toán cần giải quyết sẽ tìm ra cách lựa chọn tasks cho mỗi worker sao cho chi phí tối ưu nhất tương ứng với năng lực và tỉ lệ phân bổ.





Hình 2. 1 Bài toán Assignment

## **Lựa chọn mô hình cho bài toán**

Trong bối cảnh giải quyết các bài toán phân bổ (Assignment problems), Google Developer đề xuất sử dụng hai mô hình chính là MIP và CP-SAT. Cả hai mô hình này đều có những đặc điểm và lợi thế riêng, phù hợp với từng loại bài toán cụ thể.

a) MIP

MIP là một loại bài toán tối ưu hóa của SCIP mà trong đó các biến có thể là số nguyên hoặc số thực, và hàm mục tiêu cũng như các ràng buộc đều là tuyến tính. MIP sử dụng các kỹ thuật như nhánh và cận (branch-and-bound), mặt phẳng cắt (cutting planes) để giải quyết bài toán. Các bài toán được biểu diễn dưới dạng ma trận với các ràng buộc và hàm mục tiêu tuyến tính.

Về ưu điểm, MIP rất mạnh mẽ trong việc giải các bài toán tuyến tính với các ràng buộc số nguyên. Có khả năng cung cấp các lời giải tối ưu với bằng chứng toán học về độ chính xác.

Về nhược điểm, hiệu suất giải quyết các bài toán lớn và phức tạp có thể bị hạn chế do kích thước và tính phức tạp của mô hình. Thời gian giải có thể tăng rất nhanh khi số biến và số ràng buộc tăng lên, đặc biệt với các biến số nguyên.

b) CP-SAT

CP-SAT kết hợp giữa lập trình ràng buộc (Constraint Programming) và giải quyết bài toán thỏa mãn Boolean (SAT). CP-SAT sử dụng mô hình ràng buộc để biểu diễn bài toán và các kỹ thuật từ SAT solving để tìm kiếm lời giải. CP tập trung vào biểu diễn bài toán dưới dạng các ràng buộc và tìm kiếm trong không gian ràng buộc, sử dụng các kỹ thuật như propagation (lan truyền), backtracking (quay lui) và các hàm Heuristic để tối ưu.

Về ưu điểm, CP-SAT rất hiệu quả trong việc giải quyết các bài toán tổ hợp phức tạp với nhiều ràng buộc phi tuyến tính hoặc phi tuyến tính tổng quát. Có khả năng mô hình hóa các ràng buộc phức tạp và linh hoạt hơn so với MIP. Hiệu quả trong việc tìm kiếm các lời giải khả thi nhanh chóng.

Về nhược điểm, có thể không mạnh mẽ bằng MIP trong việc tìm kiếm lời giải tối ưu với các bài toán tuyến tính số nguyên lớn. Khó khăn hơn trong việc cung cấp bằng chứng toán học về độ chính xác của lời giải.

Như vậy, ta có bảng so sánh tóm tắt như sau:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **MIP** | **CP-SAT** |
| Phương pháp giải | Tuyến tính, nhánh và cận, mặt phẳng cắt | Ràng buộc, SAT solving, lan truyền, quay lui, heuristic |
| Mô hình hóa | Dễ dàng với các bài toán tuyến tính | Linh hoạt với các ràng buộc phi tuyến tính |
| Ứng dụng | Bài toán tuyến tính số nguyên | Bài toán tổ hợp phức tạp |
| Công cụ | SCIP (Solving Constraint Integer Programs) | CP-SAT (Constraint Programming with SAT solving) |
| Hiệu suất | Mạnh mẽ với bài toán tuyến tính nhỏ | Tốt với bài toán có ràng buộc phức tạp |

Bảng 2. 1 So sánh mô hình MIP và CP-SAT

Việc lựa chọn giữa MIP và CP-SAT phụ thuộc vào bản chất của bài toán cần giải. Nếu bài toán chủ yếu là tuyến tính và có số lượng lớn biến số nguyên, MIP có thể là lựa chọn tốt hơn. Ngược lại, nếu bài toán có nhiều ràng buộc phức tạp và phi tuyến tính, CP-SAT có thể cung cấp hiệu suất tốt hơn. Trong nhiều trường hợp, sự kết hợp giữa hai phương pháp này cũng có thể mang lại kết quả tối ưu nhất. [11]

Đối với bài toán của đề tài, sự xuất hiện của các ràng buộc phức tạp không phải là ít, do đó CP-SAT được chọn để sử dụng giải quyết các bài toán này. CP-SAT sẽ giúp xử lý hiệu quả các ràng buộc phức tạp và tìm kiếm các giải pháp phù hợp một cách tối ưu.

## **Bài toán tìm nhà cung cấp xe phù hợp**

### Phân tích bài toán

Để tìm được một nhà cung cấp xe đủ điều kiện đáp ứng cho một tuyến, cần xem xét kỹ lưỡng các tiêu chí sau. Đầu tiên, cần kiểm tra thời gian bắt đầu và kết thúc của tuyến, đảm bảo rằng thời gian này nằm trong khoảng thời gian phục vụ của loại xe mà nhà cung cấp đó đáp ứng. Mỗi loại xe sẽ có khung giờ phục vụ riêng, và điều này phải phù hợp với lịch trình của tuyến để đảm bảo vận chuyển diễn ra suôn sẻ và đúng giờ. Tiếp theo, phải đánh giá khả năng cung cấp dịch vụ của nhà cung cấp dựa trên nguồn lực hiện có, đảm bảo họ có đủ số lượng xe để đáp ứng số lượng tuyến trong ngày. Trong số các nhà cung cấp có thể đáp ứng yêu cầu về thời gian và nguồn lực, ưu tiên sẽ được dành cho những nhà cung cấp có giá thuê rẻ hơn, giúp giảm chi phí vận hành và tối ưu hóa ngân sách. Cuối cùng, tổng chi phí khi thuê đủ nhà cung cấp cho các tuyến phải được tối ưu hóa, bao gồm giá thuê xe và các chi phí phát sinh như bảo dưỡng, nhiên liệu và các chi phí khác. Một phân tích tổng thể về chi phí sẽ giúp chọn được nhà cung cấp cung cấp dịch vụ vận chuyển với mức giá hợp lý nhất mà vẫn đảm bảo chất lượng và hiệu quả. Như vậy, lựa chọn nhà cung cấp xe phù hợp đòi hỏi một quy trình đánh giá toàn diện, đảm bảo rằng các tuyến vận chuyển sẽ được thực hiện một cách hiệu quả và tiết kiệm nhất.

### Giải quyết bài toán sử dụng ngôn ngữ Java

Khởi tạo dữ liệu cho bài toán. Bao gồm ma trận chi phí costs, mảng totalSizeMax, ma trận thời gian times và ma trận thời gian workerShift. Mỗi hàng trong ma trận costs đại diện cho một nhà cung cấp và mỗi cột đại diện cho một tuyến. Mảng totalSizeMax chứa số tuyến có thể đáp ứng của mỗi nhà cung cấp. Mỗi hàng trong ma trận times chứa hai giá trị thời gian bắt đầu và kết thúc cho mỗi tuyến. Mỗi hàng trong ma trận workerShift chứa hai giá trị thời gian bắt đầu và kết thúc nằm ngoài ca làm việc của nhà cung cấp.

int[][] costs = {

                { 90, 76, 75, 20 },

                { 35, 85, 55, 20 },

                { 35, 85, 55, 20 },

};

int[][] times = { { 0, 10 }, { 11, 20 }, { 19, 23 }, { 19, 23 } };

int[][][] workerShift = {

{ { 0, 4 } }, { { 20, 23 } }, { { 0, 10 }, { 11, 18 } }

};

int[] totalSizeMax = { 1, 2, 1 };

final int numWorkers = costs.length;

final int numTasks = costs[0].length;

final int[] allWorkers = IntStream.range(0, numWorkers).toArray();

final int[] allTasks = IntStream.range(0, numTasks).toArray();

Tiếp theo khởi tạo một đối tượng CpModel, đại diện cho mô hình của bài toán. Đối tượng CpModel là nơi mà ta định nghĩa các biến, ràng buộc và hàm mục tiêu cho bài toán tối ưu hóa ràng buộc. Bằng cách sử dụng đối tượng này, ta có thể thêm các biến nhị phân (boolean), biến số nguyên, các ràng buộc, và hàm mục tiêu vào mô hình.

Khi một đối tượng CpModel được khởi tạo, ta có thể sử dụng các phương thức của nó để thêm và quản lý các thành phần của bài toán tối ưu hóa ràng buộc, từ đó tạo ra một mô hình hoàn chỉnh để giải quyết bài toán.

CpModel model = new CpModel();

Khởi tạo các biến quyết định. Trong bài toán này, mỗi biến x[worker][task] là một biến nhị phân (boolean) đại diện cho tuyến task có được giao cho nhà cung cấp worker hay không.

Literal[][] x = new Literal[numWorkers][numTasks];

        for (int worker : allWorkers) {

            for (int task : allTasks) {

x[worker][task] = model.newBoolVar(“x[“ + worker + “,” + task + “]”);

            }

        }

Sau khi đã có model và biến, ta sẽ tạo các rảng buộc cho bài toán:

a) Ràng buộc về thời gian, đảm bảo thời gian của tuyến phải nằm ngoài khung thời gian của nhà cung cấp không đáp ứng.

// Additional variables and constraints for task scheduling

        for (int worker : allWorkers) {

            List<IntervalVar> lst = new ArrayList<>();

            // shift constraint

            if (workerShift[worker] != null) {

                for (int[] shift : workerShift[worker]) {

                    int shift\_st = shift[0];

                    int shift\_en = shift[1];

                    IntVar \_shift\_st = model.newConstant(shift\_st);

                    IntVar \_shift\_en = model.newConstant(shift\_en);

                    IntVar \_shift\_size = model.newConstant(shift\_en - shift\_st);

                    lst.add(model.newIntervalVar(\_shift\_st, \_shift\_size, \_shift\_en,

                            “worker-task-shift” + worker));

                }

            }

            // time constraint

            for (int task : allTasks) {

                int st = times[task][0];

                int en = times[task][1];

                IntVar \_st = model.newConstant(st);

                IntVar \_en = model.newConstant(en);

                IntVar \_size = model.newConstant(en - st);

                lst.add(model.newOptionalIntervalVar(\_st, \_size, \_en, x[worker][task],

                        “worker-task” + task + worker));

            }

model.addNoOverlap(lst);

}

Cụ thể, vòng lặp đầu tiên for (int worker : allWorkers) duyệt qua tất cả các nhà cung cấp có trong danh sách allWorkers. Trong mỗi vòng lặp này, một danh sách lst được khởi tạo để chứa các biến thời gian của các tuyến mà nhà cung cấp đang xem xét.

Trong vòng lặp lồng bên trong for (int[] shift : workerShift[worker]), Mỗi khung thời gian làm việc của nhà cung cấp được xác định bởi thời gian bắt đầu (shift\_st) và thời gian kết thúc (shift\_en) từ mảng workerShift. Ba biến số nguyên \_shift\_st, \_shift\_en, và \_shift\_size được tạo bằng cách sử dụng phương thức model.newConstant() để đại diện cho thời gian bắt đầu, thời gian kết thúc và thời lượng của mỗi khung thời gian làm việc.

Sau đó, tạo biến thời gian cho mỗi khung thời gian làm việc của nhà cung cấp bằng cách sử dụng phương thức model.newIntervalVar(). Mỗi biến thời gian này đại diện cho một khoảng thời gian mà nhà cung cấp không thể thực hiện các tuyến.

Tương tự, trong vòng lặp lồng bên trong for (int task : allTasks), mỗi tuyến task trong danh sách allTasks được xem xét. Đầu tiên, thời gian bắt đầu (st) và thời gian kết thúc (en) của tuyến được lấy từ ma trận times. Ba biến số nguyên \_st, \_en, và \_sten được khởi tạo bằng cách sử dụng phương thức model.newConstant() để đại diện cho thời gian bắt đầu, kết thúc và thời lượng của tuyến tương ứng.

Cuối cùng, một biến thời gian tùy chọn (optional interval variable) được tạo bằng cách sử dụng phương thức model.newOptionalIntervalVar(). Biến này chứa thông tin về thời gian bắt đầu (\_st), thời lượng (\_sten), và thời gian kết thúc (\_en) của tuyến, cũng như một biến nhị phân x[worker][task] để đại diện cho việc giao tuyến task cho nhà cung cấp worker. Chuỗi “worker-task” được sử dụng để gán một tên cho biến.

Sự khác nhau giữa newIntervalVar và newOptionalIntervalVar đó là phương thức model.newIntervalVar() tạo ra các biến thời gian bắt buộc, trong khi model.newOptionalIntervalVar() tạo ra các biến thời gian tùy chọn, cho phép công việc không được lên lịch trong một khung thời gian cụ thể.

Sau khi tạo biến thời gian cho tất cả các tuyến và thời gian của nhà cung cấp, ràng buộc model.addNoOverlap(lst) được thêm vào mô hình để đảm bảo rằng mỗi tuyến chỉ được lên lịch trong đúng khung thời gian làm việc của nhà cung cấp.

b) Ràng buộc về khả năng đáp ứng số tuyến của nhà cung cấp.

for (int worker : allWorkers) {

List<Literal> workerTasks = new ArrayList<>();

for (int task : allTasks) {

workerTasks.add(x[worker][task]);

}

IntVar totalTasksForWorker = model.newIntVar(0, totalSizeMax[worker], “totalTasksForWorker\_” + worker);

Literal[] assignedWorkersArray = workerTasks.toArray(new Literal[0]);

model.addEquality(LinearExpr.sum(assignedWorkersArray), totalTasksForWorker);

}

Vòng lặp for (int worker : allWorkers) lặp qua tất cả các nhà cung cấp trong danh sách allWorkers. Sau đó một danh sách workerTasks (kiểu Literal) được khởi tạo để chứa các biến biểu diễn việc phân công tuyến cho nhà cung cấp hiện tại. Vòng lặp for (int task : allTasks) lặp qua tất cả các tuyến trong danh sách allTasks. Biến x[worker][task] biểu diễn việc tuyến task được giao cho nhà cung cấp worker, biến này được thêm vào danh sách workerTasks.

Biến totalTasksForWorker được tạo ra để biểu diễn tổng số tuyến được giao cho nhà cung cấp worker. Biến này là một đối tượng IntVar có giá trị từ 0 đến totalSizeMax[worker], nơi totalSizeMax là một mảng chứa giới hạn tối đa của số tuyến cho từng nhà cung cấp. Sau đó chuyển danh sách workerTasks sang mảng Literal[] assignedWorkersArray. Và thêm ràng buộc model.addEquality(LinearExpr.sum(assignedWorkersArray),totalTasksForWorker), ràng buộc này thêm vào mô hình để đảm bảo rằng tổng số tuyến được giao cho nhà cung cấp worker (tính bằng tổng của các phần tử trong assignedWorkersArray) bằng với biến totalTasksForWorker.

c) Ràng buộc về mỗi tuyến được giao cho một nhà cung cấp.

for (int task : allTasks) {

            List<Literal> workers = new ArrayList<>();

            for (int worker : allWorkers) {

                workers.add(x[worker][task]);

            }

            model.addExactlyOne(workers);

        }

Cụ thể, vòng lặp bên ngoài for (int task : allTasks) duyệt qua tất cả các tuyến có trong danh sách allTasks. Trong mỗi vòng lặp này, một danh sách workers được khởi tạo để chứa tất cả các biến nhị phân (boolean variables) x[worker][task], đại diện cho việc gán tuyến task cho các nhà cung cấp khác nhau.

Trong vòng lặp lồng bên trong for (int worker : allWorkers), mỗi nhà cung cấp worker trong danh sách allWorkers được xem xét. Biến x[worker][task] tương ứng với việc gán tuyến task cho nhà cung cấp worker. Biến này được thêm vào danh sách workers.

Sau khi đã thêm tất cả các biến x[worker][task] tương ứng với việc gán tuyến task cho các nhà cung cấp khác nhau vào danh sách workers, ràng buộc model.addExactlyOne(workers) được thêm vào mô hình. Ràng buộc này đảm bảo rằng chính xác một biến trong danh sách workers sẽ nhận giá trị true, đồng thời các biến còn lại sẽ nhận giá trị false, đảm bảo rằng mỗi tuyến chỉ được giao cho đúng một nhà cung cấp duy nhất.

Sau khi đã có các ràng buộc, ta tiếp tục xác định hàm mục tiêu của bài toán.

// Objective

        LinearExprBuilder obj = LinearExpr.newBuilder();

        for (int worker : allWorkers) {

            for (int task : allTasks) {

                obj.addTerm(x[worker][task], costs[worker][task]);

            }

        }

        model.minimize(obj);

LinearExprBuilder obj = LinearExpr.newBuilder(). Ở đây, một đối tượng LinearExprBuilder được tạo ra để xây dựng hàm mục tiêu tuyến tính. Đối tượng này sẽ được sử dụng để thêm các thành phần của hàm mục tiêu vào.

Vòng lặp hai lớp for duyệt qua tất cả các cặp (nhà cung cấp, tuyến) có trong ma trận chi phí costs.

Vòng lặp bên ngoài for (int worker : allWorkers) duyệt qua tất cả các nhà cung cấp.

Vòng lặp lồng bên trong for (int task : allTasks) duyệt qua tất cả các tuyến.

Trong mỗi lần lặp của vòng lặp lồng, một hạng tử (term) mới được thêm vào hàm mục tiêu. Hạng tử này được tạo bằng cách sử dụng phương thức addTerm() của đối tượng obj. Hạng tử này có dạng x[worker][task] \* costs[worker][task], trong đó x[worker][task] là biến quyết định đại diện cho việc giao tuyến task cho nhà cung cấp worker, và costs[worker][task] là chi phí tương ứng với việc giao tuyến đó.

Cuối cùng, hàm mục tiêu được tạo thành là tổng tất cả các hạng tử đã được thêm vào, đại diện cho tổng chi phí của việc gán tuyến cho các nhà cung cấp. model.minimize(obj) hàm mục tiêu được thiết lập bằng cách sử dụng phương thức minimize() của đối tượng mô hình model, khẳng định rằng mục tiêu của bài toán là tối thiểu hóa giá trị của hàm mục tiêu, tức là tối thiểu hóa tổng chi phí của việc gán tuyến cho các nhà cung cấp.

Tiếp theo, ta giải quyết bài toán bằng đối tượng CpSolver.

// Solve

        CpSolver solver = new CpSolver();

        CpSolverStatus status = solver.solve(model);

Cụ thể, tạo một đối tượng CpSolver, đây là một solver được cung cấp bởi công cụ OR-Tools để giải quyết các bài toán tối ưu hóa ràng buộc dựa trên phương pháp lập trình ràng buộc.

CpSolverStatus status = solver.solve(model); Sử dụng solver để giải quyết bài toán được mô hình hóa bởi model. Ở đây thuật toán quay lui (Backtracking) được sử dụng kết hợp với các hàm Heuristic để tối ưu hóa và kỹ thuật ràng buộc (Constraint Programing) áp dụng vào các lần kiểm tra để xác định đúng với ràng buộc đã khai báo, cùng với kỹ thuật lan truyền (Propagation) sử dụng để đơn giản hóa bài toán và thu hẹp không gian tìm kiếm.

Kết quả của quá trình giải quyết được lưu trữ trong biến status, là một giá trị thuộc kiểu enum CpSolverStatus, biểu thị trạng thái của quá trình giải quyết bài toán. Nếu status là CpSolverStatus.OPTIMAL, ngụ ý rằng giải pháp tối ưu đã được tìm thấy, tức là đã tìm ra một giải pháp có giá trị của hàm mục tiêu là tối ưu nhất. Nếu status là CpSolverStatus.FEASIBLE, có nghĩa là solver đã tìm thấy một giải pháp khả thi, nhưng không chắc chắn rằng nó là tối ưu nhất. Nếu không, status có thể là CpSolverStatus.INFEASIBLE, CpSolverStatus.MODEL\_INVALID, CpSolverStatus.UNKNOWN, hoặc các giá trị khác, biểu thị rằng có lỗi trong quá trình giải hoặc mô hình không hợp lệ.

Sau khi quá trình giải quyết bài toán được thực hiện, ta tiếp tục với việc in ra giải pháp đã tìm được (nếu có) và kiểm tra xem giải pháp đó có khả thi hay không.

// Print solution.

        // Check that the problem has a feasible solution.

        if (status == CpSolverStatus.OPTIMAL || status == CpSolverStatus.FEASIBLE) {

            System.out.println(“Total cost: “ + solver.objectiveValue() + “\n”);

            for (int worker : allWorkers) {

                for (int task : allTasks) {

                    if (solver.booleanValue(x[worker][task])) {

                        System.out.println(“Worker “ + worker + “ assigned to task “ + task

                                + “.  Cost: “ + costs[worker][task]);

                    }

                }

            }

        } else {

            System.err.println(“No solution found.”);

        }

Trước tiên, một kiểm tra được thực hiện để đảm bảo rằng bài toán có một giải pháp khả thi. Nó được thực hiện bằng cách kiểm tra xem biến status có giá trị là CpSolverStatus.OPTIMAL hoặc CpSolverStatus.FEASIBLE không. Nếu có ít nhất một giải pháp tối ưu hoặc khả thi được tìm thấy, đoạn mã tiếp tục với việc in ra giải pháp.

Nếu bài toán có một giải pháp tối ưu hoặc khả thi sẽ in ra thông tin về giải pháp đó. Đầu tiên, tổng chi phí của giải pháp được in ra bằng cách sử dụng phương thức solver.objectiveValue(), đại diện cho giá trị của hàm mục tiêu tối ưu. Sau đó, vòng lặp duyệt qua tất cả các cặp (nhà cung cấp, tuyến) và kiểm tra xem biến x[worker][task] có được gán giá trị true không. Nếu có, thông tin về gán tuyến đó cho nhà cung cấp tương ứng được in ra, bao gồm thông tin về tuyến, nhà cung cấp và chi phí tương ứng. Nếu không có giải pháp nào được tìm thấy, một thông báo lỗi được in ra để thông báo rằng không có giải pháp nào được tìm thấy cho bài toán đã cho.

Qua đó, đoạn mã này không chỉ giúp hiển thị thông tin về giải pháp mà còn giúp xác nhận tính khả thi của giải pháp, đảm bảo rằng kết quả được in ra là đáng tin cậy.

Kết quả của quá trình giải quyết bài toán được hiển thị như sau:

Total cost: 195.0

Worker 0 assigned to task 3. Cost: 20

Worker 1 assigned to task 0. Cost: 35

Worker 1 assigned to task 1. Cost: 85

Worker 2 assigned to task 2. Cost: 55

Điều này có nghĩa là bài toán đã được giải quyết thành công và đã tìm thấy một giải pháp khả thi. Giải pháp này gồm các phân công tuyến cho các nhà cung cấp, với tổng chi phí là 195.0.

Cụ thể:

Nhà cung cấp 0 được giao tuyến 3 với chi phí là 20.

Nhà cung cấp 1 được giao tuyến 0 với chi phí là 35.

Nhà cung cấp 1 được giao tuyến 1 với chi phí là 85.

Nhà cung cấp 2 được giao tuyến 2 với chi phí là 55.

## **Bài toán tìm nhân viên phù hợp**

### Phân tích bài toán

Để lựa chọn một nhân viên phù hợp cho một tuyến, cần phải xem xét kỹ lưỡng nhiều tiêu chí quan trọng nhằm đảm bảo hiệu suất làm việc tối ưu và quản lý công nợ hiệu quả. Trước hết, yếu tố thời gian làm việc là rất quan trọng. Cần kiểm tra thời gian bắt đầu và kết thúc của tuyến, đảm bảo rằng khung thời gian này nằm trong ca làm việc của nhân viên được lựa chọn, giúp tránh vấn đề nhân viên phải làm việc ngoài giờ, từ đó giảm thiểu rủi ro về mặt sức khỏe và tăng cường năng suất lao động. Tiếp theo, mỗi nhân viên đều có công nợ hiện tại và hạn mức công nợ cá nhân, trong khi mỗi tuyến lại đi kèm với một khoản phí thu hộ nhất định. Khi gán nhân viên vào tuyến, cần đảm bảo rằng công nợ của họ không vượt quá hạn mức công nợ đã được quy định. Như vậy sẽ không chỉ giúp kiểm soát rủi ro tài chính mà còn đảm bảo rằng nhân viên không bị áp lực quá lớn về mặt tài chính trong quá trình làm việc. Ngoài ra, kỹ năng của nhân viên cũng là một yếu tố then chốt. Mỗi tuyến yêu cầu những kỹ năng nhất định, chẳng hạn như giao hàng hoặc giao hàng kèm theo lắp đặt. Do đó, cần đảm bảo rằng nhân viên được gán vào tuyến có đủ kỹ năng và kinh nghiệm phù hợp để thực hiện tuyến một cách hiệu quả, giúp đảm bảo chất lượng dịch vụ và tăng sự hài lòng của khách hàng. Cuối cùng, tổng công nợ của các nhân viên cho các tuyến cần được tối ưu hóa. Một phân tích tổng thể về chi phí công nợ sẽ giúp chọn ra những nhân viên có công nợ hợp lý nhất, từ đó tối ưu hóa chi phí và hiệu quả hoạt động. Việc này không chỉ giúp doanh nghiệp quản lý tài chính tốt hơn mà còn tăng cường hiệu suất làm việc của toàn bộ đội ngũ nhân viên. Bằng cách đánh giá toàn diện các yếu tố trên, ta có thể chọn ra những nhân viên phù hợp nhất cho từng tuyến, tối ưu hóa hiệu suất làm việc và quản lý công nợ một cách hiệu quả.

### Giải quyết bài toán sử dụng ngôn ngữ Java

Khởi tạo dữ liệu cho bài toán. Bao gồm ma trận chi phí công nợ costs, mảng skillRequire, ma trận kỹ năng workerSkill, mảng max\_debt, ma trận thời gian times và ma trận thời gian workerShift. Mỗi hàng trong ma trận costs đại diện cho một nhân viên và mỗi cột đại diện cho một tuyến. Mảng skillRequire chứa kỹ năng yêu cầu của mỗi tuyến. Mỗi hàng trong ma trận workerSkill chứa các kỹ năng của một nhân viên. Mỗi hàng trong mảng max\_debt chứa một giá trị hạn mức công nợ của nhân viên. Mỗi hàng trong ma trận times chứa hai giá trị thời gian bắt đầu và kết thúc cho mỗi tuyến. Mỗi hàng trong ma trận workerShift chứa hai giá trị thời gian bắt đầu và kết thúc nằm ngoài ca làm việc của nhà cung cấp.

// Data

        int[][] costs = {

                { 120, 150, 80 },

                { 150, 225, 90 },

                { 156, 195, 104 },

        };

        int[] skillRequire = { 1, 1, 1 };

        int[][] workerSkill = {

                { 1, 2, 3 }, // worker 1

                { 1, 2, 3 }, // worker 2

                { 1, 2, 3 }, // worker 3

        };

        int[] max\_debt = { 300, 100, 500 };

        int[][] times = { { 0, 10 }, { 11, 20 }, { 5, 20 } };

        int[][][] workerShift = {

                { { 0, 4 } },

                { { 10, 11 } },

                { { 10, 11 } }

        };

        final int numWorkers = costs.length;

        final int numTasks = costs[0].length;

        final int[] allWorkers = IntStream.range(0, numWorkers).toArray();

        final int[] allTasks = IntStream.range(0, numTasks).toArray();

Các bước từ tạo đối tượng CpModel, tạo các biến quyết định, hàm mục tiêu, và ràng buộc về thời gian tuyến với mỗi nhân viên đều tương tự như tìm nhà cung cấp phù hợp với tuyến ở mục 2.3.2 nên phần tiếp theo chỉ tập trung mô tả các bước còn lại.

a) Ràng buộc về công nợ hiện tại của nhân viên khi được gán vào các tuyến phải nhỏ hơn hạn mức công nợ tương ứng.

for (int worker : allWorkers) {

            LinearExprBuilder debt\_lst = LinearExpr.newBuilder();

            for (int task : allTasks) {

                debt\_lst.addTerm(x[worker][task], costs[worker][task]);

            }

            model.addLessOrEqual(debt\_lst, max\_debt[worker]);

        }

Vòng lặp đầu tiên duyệt qua tất cả các phần tử của allWorkers sau đó tạo một đối tượng LinearExprBuilder mới có tên là debt\_lst. LinearExprBuilder là một lớp được sử dụng để xây dựng các biểu thức tuyến tính. LinearExpr.newBuilder() là một phương thức tĩnh trả về một đối tượng LinearExprBuilder mới. Vòng lặp tiếp theo duyệt qua tất cả các phần tử của allTasks sau đó thêm một hạng tử vào biểu thức tuyến tính debt\_lst. Với [worker][task] là một biến quyết định biểu diễn việc nhân viên worker có được gán cho tuyến task hay không và costs[worker][task] là chi phí tương ứng nếu nhân viên worker thực hiện tuyến task.

Thêm ràng buộc “addLessOrEqual” vào mô hình model. Ràng buộc này đảm bảo rằng biểu thức tuyến tính debt\_lst (tổng công nợ mà nhân viên worker phải gánh) không vượt quá max\_debt[worker] là giá trị tối đa công nợ mà nhân viên worker có thể chấp nhận.

Như vậy, ràng buộc đảm bảo rằng tổng công nợ mà mỗi nhân viên phải gánh không vượt quá một giá trị tối đa cụ thể. Mỗi nhân viên có một biểu thức tuyến tính đại diện cho tổng công nợ của các nhiệm vụ mà họ được gán, và biểu thức này được so sánh với giá trị tối đa cho phép.

b) Ràng buộc về kỹ năng nhân viên được gán phải phù hợp với kỹ năng yêu cầu của tuyến.

for (int task : allTasks) {

            for (int worker : allWorkers) {

                for (int skill : workerSkill[worker]) {

                    if (skillRequire[task] == skill) {

                        IntVar skillRequirementVar = model.newConstant(skillRequire[task]);

                        IntVar workerSkillVar = model.newConstant(skill);

                        model.addEquality(skillRequirementVar, workerSkillVar).onlyEnforceIf(x[worker][task]);

                    }

                }

            }

        }

Vòng lặp đầu tiên này duyệt qua tất cả các tuyến (task) trong danh sách allTasks. Mỗi task đại diện cho một tuyến cần được phân công. Bên trong vòng lặp tuyến, vòng lặp tiếp theo duyệt qua tất cả các nhân viên (worker) trong danh sách allWorkers. Mỗi worker đại diện cho một nhân viên có thể được phân công công việc. Bên trong vòng lặp nhân viên, vòng lặp này duyệt qua tất cả các kỹ năng (skill) mà một nhân viên cụ thể (worker) có trong danh sách workerSkill[worker]. workerSkill là một mảng hoặc danh sách mà mỗi phần tử là một tập hợp các kỹ năng mà nhân viên đó có.

Câu lệnh điều kiện “skillRequire[task] == skill” kiểm tra xem kỹ năng yêu cầu của tuyến (skillRequire[task]) có khớp với kỹ năng hiện tại của nhân viên (skill) hay không. skillRequire là một mảng hoặc danh sách lưu trữ kỹ năng yêu cầu cho mỗi tuyến. Tiếp theo tạo một biến hằng số (IntVar) trong mô hình tối ưu hóa để đại diện cho kỹ năng yêu cầu của tuyến hiện tại, và tạo một biến hằng số (IntVar) trong mô hình tối ưu hóa để đại diện cho kỹ năng của nhân viên hiện tại.

Cuối cùng, thêm ràng buộc addEquality(skillRequirementVar, workerSkillVar).onlyEnforceIf(x[worker][task]) vào mô hình tối ưu hóa để yêu cầu rằng kỹ năng của nhân viên phải khớp với kỹ năng yêu cầu của tuyến. Ràng buộc này chỉ được thực thi nếu nhân viên (worker) được phân công tuyến (task), được biểu diễn bằng biến nhị phân x[worker][task]. Cụ thể, model.addEquality(skillRequirementVar, workerSkillVar) sẽ tạo một ràng buộc yêu cầu hai biến (skillRequirementVar và workerSkillVar) phải bằng nhau và onlyEnforceIf(x[worker][task]) ý nghĩa là chỉ được áp dụng nếu biến nhị phân x[worker][task] có giá trị true (tức là nhân viên worker được phân công tuyến task).

Như vậy đã có đủ các ràng buộc, bước xác định hàm mục tiêu và giải bài toán cũng tương tự như tìm nhà cung cấp phù hợp ở phần 2.3.2, nên ta tiếp tục tới đoạn in kết quả.

if (status == CpSolverStatus.OPTIMAL || status == CpSolverStatus.FEASIBLE) {

            System.out.println(“Total cost: “ + solver.objectiveValue() + “\n”);

            for (int worker : allWorkers) {

                for (int task : allTasks) {

                    if (solver.booleanValue(x[worker][task])) {

                        System.out.println(“Worker “ + worker + “ assigned to task “ + task

                                + “.  Cost: “ + costs[worker][task]);

                    }

                }

            }

        } else {

            System.err.println(“No solution found.”);

        }

Trước tiên, thực hiện kiểm tra xem biến status có giá trị là CpSolverStatus.OPTIMAL hoặc CpSolverStatus.FEASIBLE không. Nếu có ít nhất một giải pháp tối ưu hoặc khả thi được tìm thấy, đoạn mã tiếp tục với việc in ra giải pháp.

Nếu bài toán có một giải pháp tối ưu hoặc khả thi sẽ in ra thông tin về giải pháp đó. Đầu tiên, tổng chi phí của giải pháp được in ra bằng cách sử dụng phương thức solver.objectiveValue(), đại diện cho giá trị của hàm mục tiêu tối ưu. Sau đó, vòng lặp duyệt qua tất cả các cặp (nhân viên, tuyến) và kiểm tra xem biến x[worker][task] có được gán giá trị true không. Nếu có, thông tin về gán tuyến đó cho nhân viên tương ứng được in ra, bao gồm thông tin về tuyến, nhan viên và công nợ tương ứng. Nếu không có giải pháp nào được tìm thấy, một thông báo lỗi được in ra để thông báo rằng không có giải pháp nào được tìm thấy cho bài toán đã cho.

Kết quả của quá trình giải quyết bài toán được hiển thị như sau:

Total cost: 431.0

Worker 0 assigned to task 2. Cost: 80

Worker 2 assigned to task 0. Cost: 156

Worker 2 assigned to task 1. Cost: 195

Điều này có nghĩa là bài toán đã được giải quyết thành công và đã tìm thấy một giải pháp khả thi. Giải pháp này gồm các phân công tuyến cho các nhân viên, với tổng công nợ là 431.0.

Cụ thể:

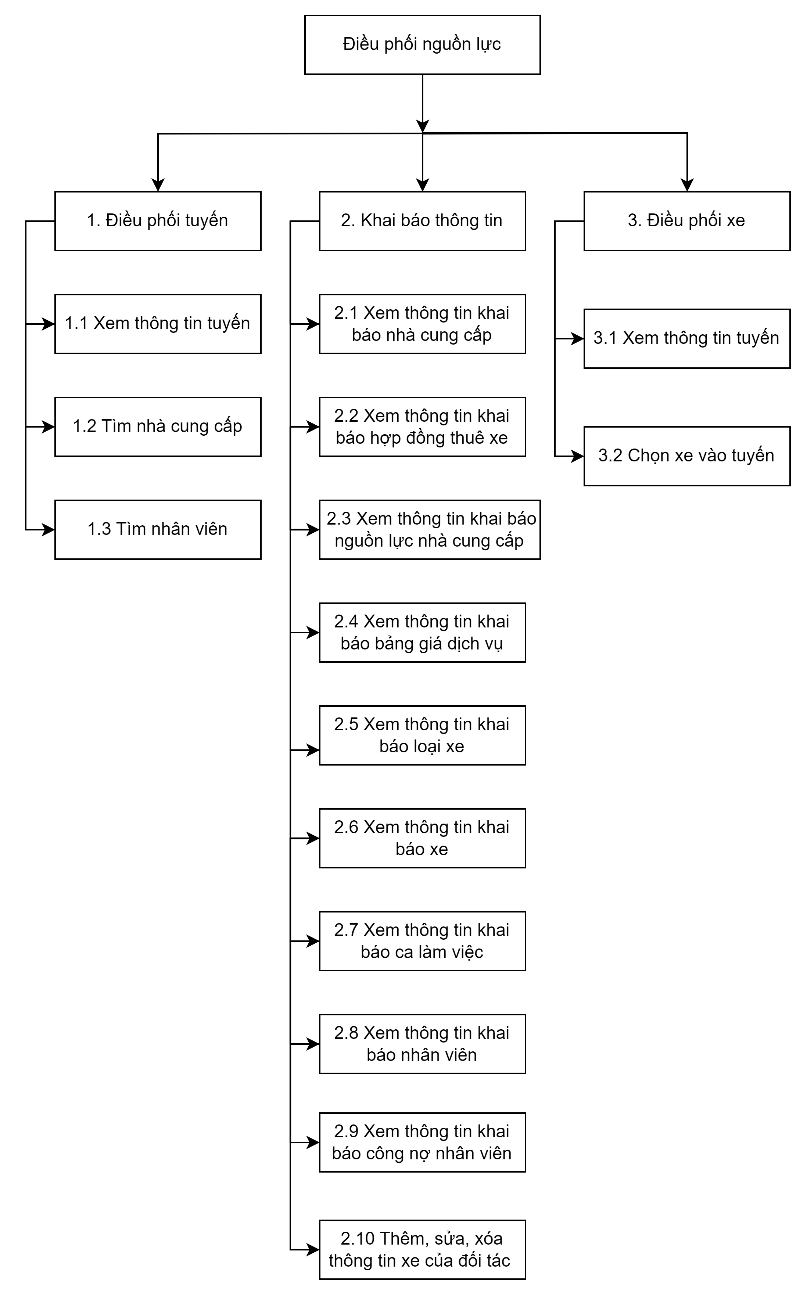
Nhân viên 0 được giao tuyến 2 với công nợ là 80.

Nhân viên 2 được giao tuyến 0 với công nợ là 156.

Nhân viên 2 được giao tuyến 1 với công nợ là 195.

# CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## **Sơ đồ phân rã chức năng – BFD**



Sơ đồ 3. 1 Sơ đồ phân rã chức năng

Sơ đồ BFD mô tả các chức năng quản lý và điều phối nguồn lực, bao gồm ba chức năng chính: Điều phối tuyến, Khai báo thông tin, và Điều phối xe. Dưới mỗi chức năng chính, có các chức năng phụ chi tiết như sau.

Chức năng đầu tiên “Điều phối tuyến”. Trong đó, có ba chức năng phụ: Xem thông tin tuyến (1.1), cho phép người dùng xem thông tin về các tuyến; Tìm nhà cung cấp (1.2), cho phép tìm kiếm nhà cung cấp xe phù hợp và Tìm nhân viên (1.3), cho phép tìm kiếm nhân viên phù hợp cho tuyến.

Chức năng thứ hai “Khai báo thông tin”. Chức năng này bao gồm nhiều chức năng phụ khác nhau: Xem thông tin khai báo nhà cung cấp (2.1) cho phép xem thông tin khai báo từ nhà cung cấp xe và nhân viên; Xem thông tin khai báo hợp đồng thuê xe (2.2) cho phép xem thông tin các hợp đồng thuê xe; Xem thông tin khai báo nguồn lực nhà cung cấp (2.3) cho phép xem thông tin về nguồn lực từ nhà cung cấp; Xem thông tin khai báo bảng giá dịch vụ (2.4) cho phép xem thông tin về bảng giá dịch vụ; Xem thông tin khai báo loại xe (2.5) cho phép xem thông tin về các loại xe; Xem thông tin khai báo xe (2.6) cho phép xem thông tin chi tiết về từng xe; Xem thông tin khai báo ca làm việc (2.7) cho phép xem thông tin về ca làm việc của nhà cung cấp và nhân viên; Xem thông tin khai báo nhân viên (2.8) cho phép xem thông tin chi tiết về nhân viên; Xem thông tin khai báo công nợ nhân viên (2.9) cho phép xem thông tin về công nợ của nhân viên; và cuối cùng là chức năng Thêm, sửa, xóa thông tin xe của đối tác (2.10) cho phép thêm mới, chỉnh sửa hoặc xóa thông tin về xe của đối tác.

Chức năng thứ ba “Điều phối xe”, bao gồm hai chức năng phụ: Xem thông tin tuyến (3.1) cho phép xem thông tin chi tiết về các tuyến đã được điều phối trước đó; và Chọn xe vào tuyến (3.2) cho phép chọn xe phù hợp để chạy trên các tuyến đã được định trước. [12]

Sơ đồ mô tả chi tiết cách các chức năng điều phối và quản lý nguồn lực được triển khai và liên kết với nhau, giúp người dùng dễ dàng hình dung và thực hiện các quy trình quản lý một cách hiệu quả.

## **Sơ đồ luồng dữ liệu – DFD**

### Mức ngữ cảnh



Sơ đồ 3. 2 Mức ngữ cảnh (Mức 0)

Sơ đồ DFD mức ngữ cảnh trình bày cách hệ thống tương tác với các bên liên quan ở cấp độ cao nhất. Sau đây là mô tả chi tiết của sơ đồ này.

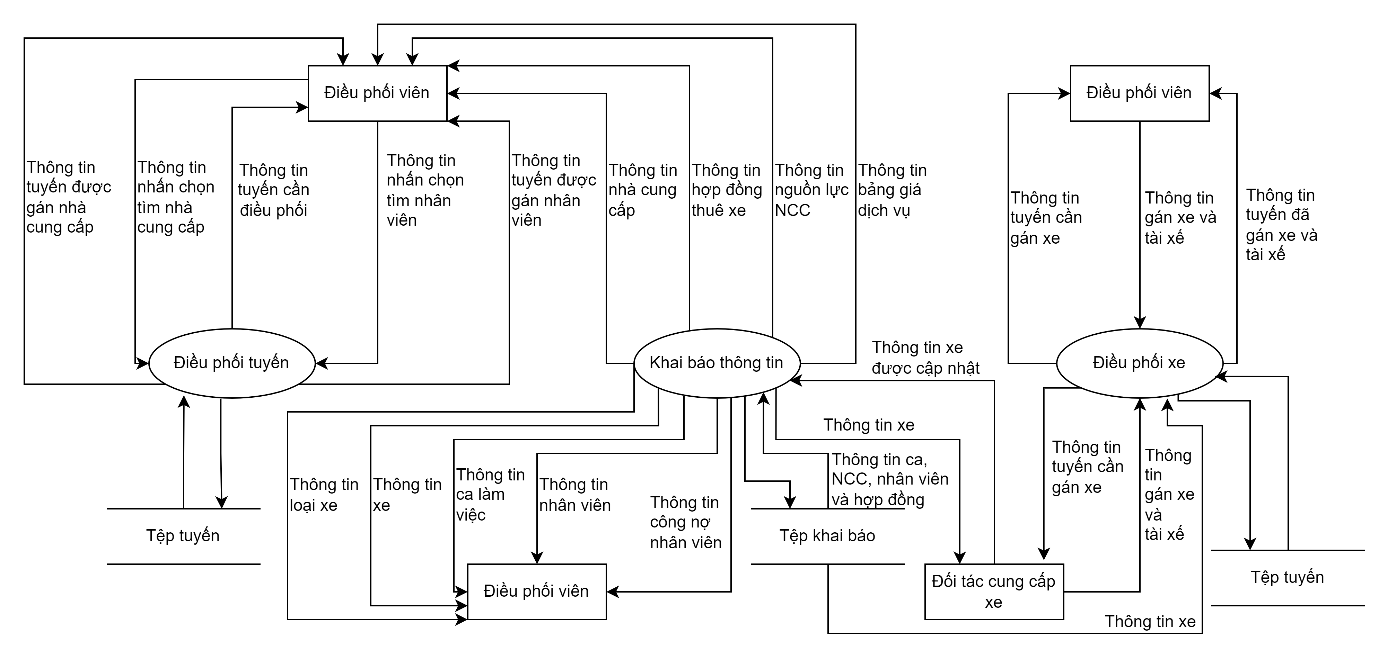
Hệ thống chính, được biểu diễn bằng một hình tròn, là quy trình điều phối nguồn lực. Đây là quy trình trung tâm của hệ thống. Có hai thực thể bên ngoài tương tác với hệ thống: điều phối viên và đối tác cung cấp xe.

Điều phối viên là nhân viên nội bộ có nhiệm vụ quan trọng. Họ kiểm tra thông tin về các tuyến cần điều phối trong ngày cùng với các dữ liệu liên quan đến nhà cung cấp và nhân viên. Sau đó, họ chọn tìm kiếm nhà cung cấp và nhân viên phù hợp cho từng tuyến. Hệ thống sẽ tự động xử lý và trả về thông tin về nhà cung cấp và nhân viên tối ưu nhất. Sau khi các tuyến đã được gán nhà cung cấp, điều phối viên tiếp tục phân bổ xe vào các tuyến được giao cho công ty nội bộ.

Đối tác cung cấp xe là các nhà cung cấp xe nhỏ lẻ. Hệ thống cho phép họ khai báo thông tin về xe của mình và phản hồi nhanh chóng các yêu cầu thuê xe từ hệ thống. Cụ thể, họ sẽ nhận được thông tin về các tuyến cần gán xe và có thể chọn xe của mình cho các tuyến đó. [12]

Với sự phân chia nhiệm vụ và quyền hạn rõ ràng, hệ thống này giúp tối ưu hóa quá trình điều phối nguồn lực, đảm bảo sự hiệu quả và linh hoạt trong quản lý và vận hành.

### Mức đỉnh



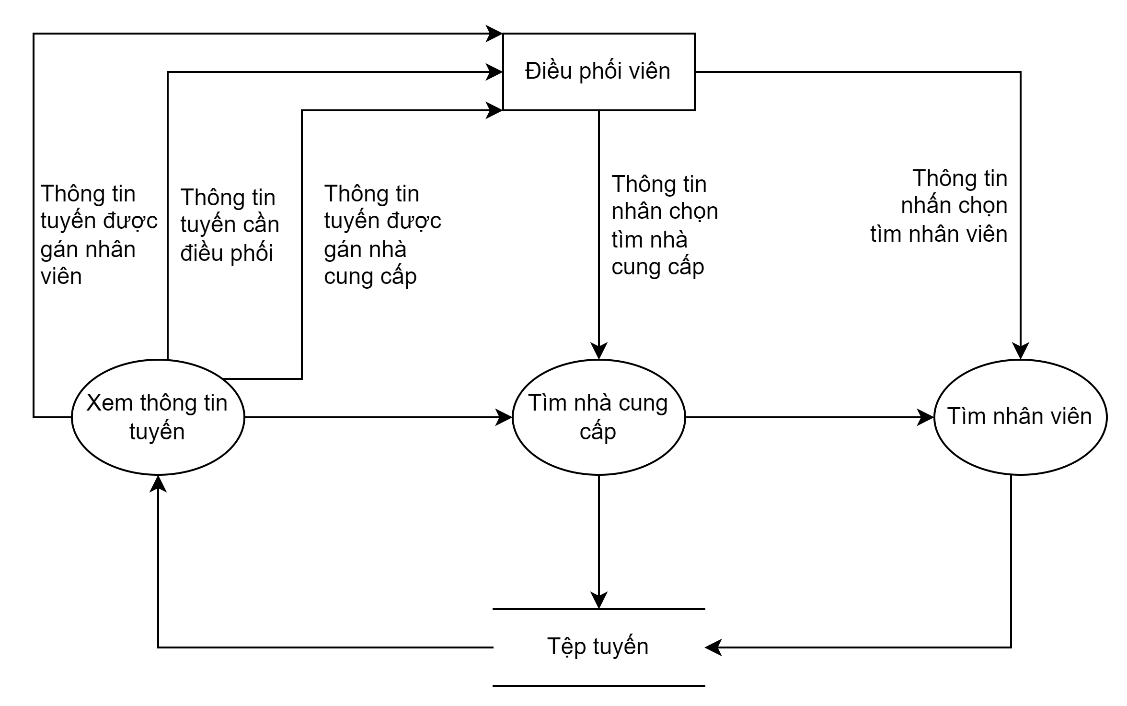
Sơ đồ 3. 3 Mức đỉnh (Mức 1)

Sơ đồ DFD mức đỉnh biểu diễn các quy trình trong hệ thống quản lý điều phối xe bao gồm các thực thể bên ngoài và các quy trình chính. Đầu tiên, hệ thống gửi thông tin các tuyến cần điều phối đến điều phối viên, điều phối viên gửi yêu cầu đến điều phối tuyến bao gồm (tìm nhà cung cấp và tìm nhân viên). Sau khi xử lý, quy trình này tạo ra tuyến điều phối đã được gán thông tin và chuyển nó đến điều phối viên. Quy trình Khai báo thông tin để lấy dữ liệu khai báo cần thiết và chuyển đến cho điều phối viên.

Bên cạnh đó, tuyến đã được gán nhà cung cấp và cần điều phối xe sẽ gửi thông tin đến điều phối viên cùng với đối tác cung cấp xe. Ở quy trình Điều phối xe, họ sẽ gửi thông tin xe cùng tài xế, quy trình sẽ nhận và xử lý các yêu cầu này cùng với thông tin khai báo từ quy trình Khai báo thông tin. Sau khi xử lý, quy trình Điều phối xe sẽ gửi thông tin điều phối xe trở lại cho điều phối viên. Như vậy, sơ đồ này mô tả rõ ràng cách quản lý và điều phối trong hệ thống thông qua các quy trình và luồng dữ liệu liên quan, đảm bảo rằng tất cả các yêu cầu và thông tin cần thiết được xử lý một cách hợp lý và hiệu quả. [12]

### Mức dưới đỉnh

#### Điều phối tuyến

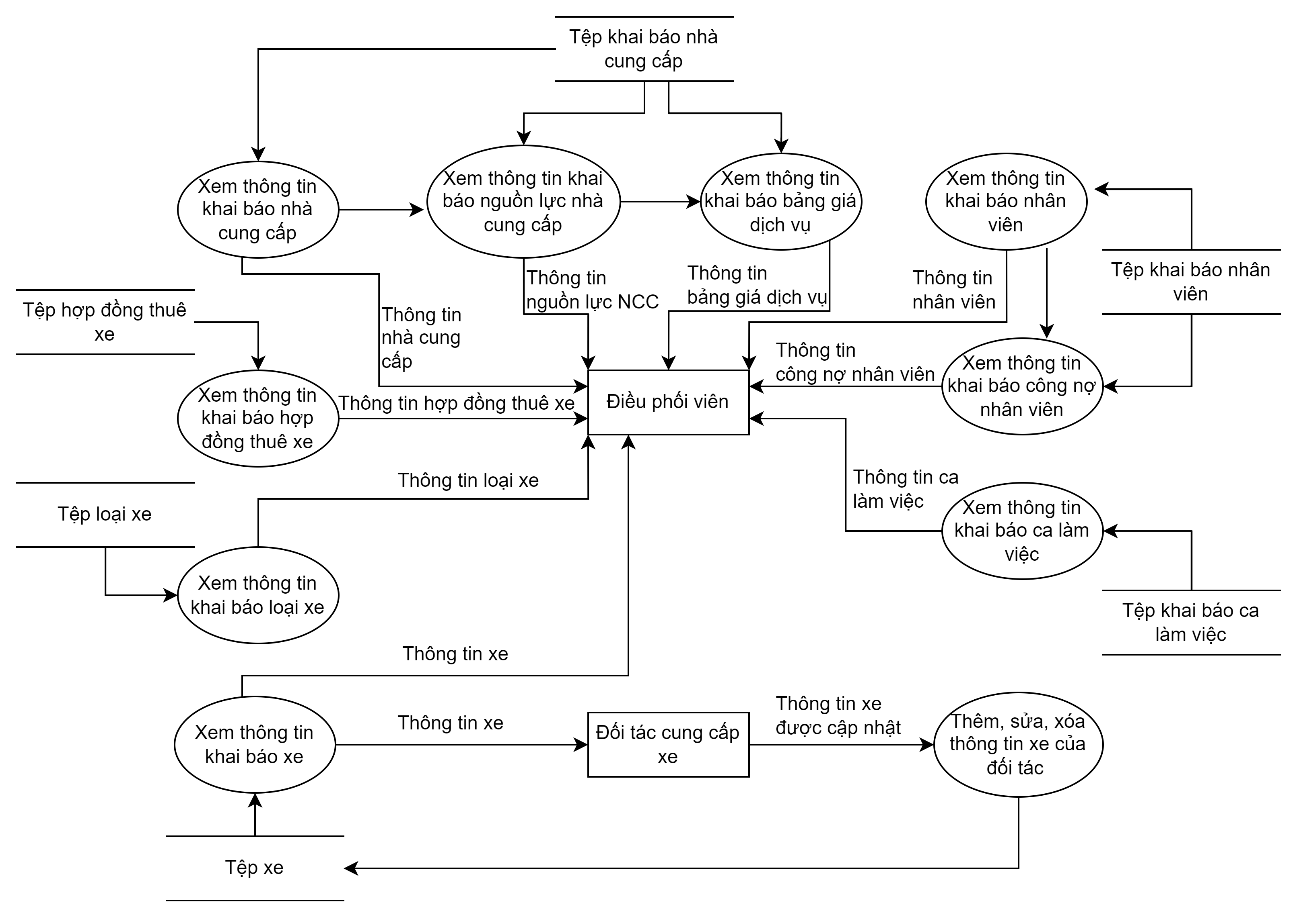


Sơ đồ 3. 4 Mức dưới đỉnh - Điều phối tuyến

Sơ đồ DFD mức dưới đỉnh của chức năng Điều phối tuyến mô tả chi tiết quy trình Điều phối tuyến trong hệ thống. Trong sơ đồ này, điều phối viên là thực thể bên ngoài gửi yêu cầu điều phối tuyến vào hệ thống. Yêu cầu này được xử lý bởi quy trình con Xem thông tin tuyến, nơi thông tin tuyến được xem xét và sau đó gửi đến hai quy trình con khác là Tìm nhà cung cấp và Tìm nhân viên. Quy trình con Tìm nhà cung cấp nhận thông tin tuyến và tiến hành tìm kiếm nhà cung cấp phù hợp, sau đó gửi thông tin nhà cung cấp trở lại quy trình Điều phối tuyến. Tương tự, quy trình con Tìm nhân viên cũng nhận thông tin tuyến và tìm kiếm nhân viên phù hợp, sau đó gửi thông tin nhân viên trở lại quy trình Điều phối tuyến. [12]

Toàn bộ quá trình này tạo ra một luồng dữ liệu liên tục giữa các quy trình con để đảm bảo rằng tất cả các thông tin cần thiết được xử lý và cung cấp một cách hợp lý và hiệu quả, giúp hệ thống điều phối tự động hoạt động mượt mà, đồng thời đảm bảo rằng việc điều phối tuyến được thực hiện chính xác và kịp thời.

#### Khai báo thông tin



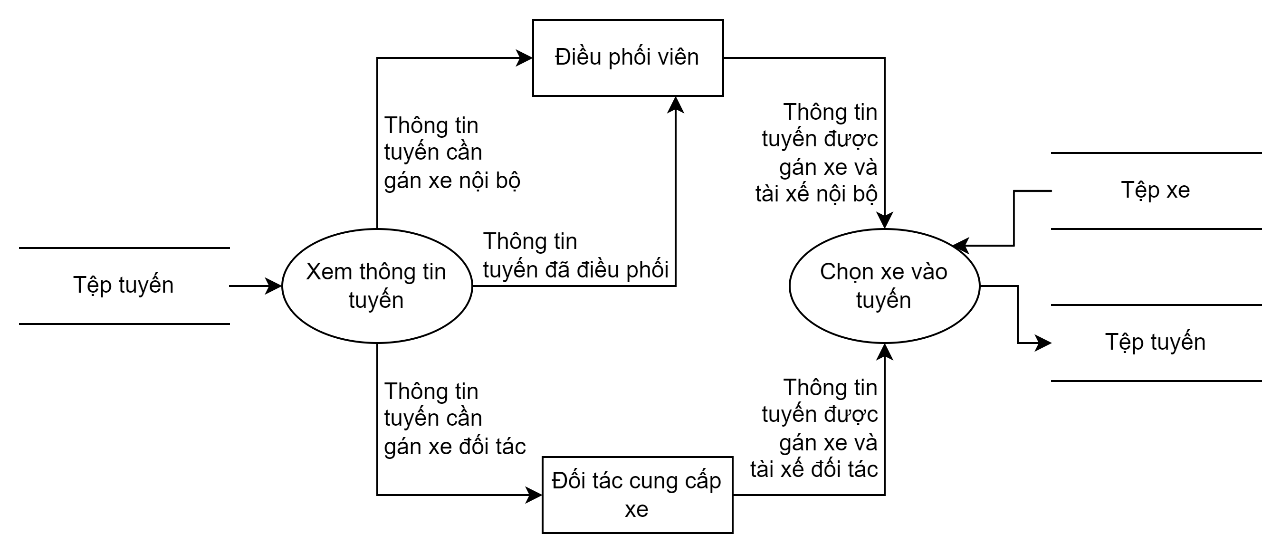
Sơ đồ 3. 5 Mức dưới đỉnh - Khai báo thông tin

Sơ đồ DFD mức dưới đỉnh của chức năng “khai báo thông tin” chi tiết hóa các tác vụ mà hệ thống thực hiện khi điều phối viên và đối tác cung cấp xe tương tác với hệ thống. Trung tâm của sơ đồ là điều phối viên, người sử dụng chính, có khả năng truy cập và xử lý các yêu cầu thông tin. Các chức năng cụ thể bao gồm: xem thông tin khai báo nhà cung cấp, cho phép điều phối viên truy cập thông tin về các nhà cung cấp; xem thông tin khai báo nguồn lực nhà cung cấp, truy cập thông tin về nguồn lực mà nhà cung cấp có thể cung cấp; và xem thông tin khai báo bảng giá dịch vụ, cho phép điều phối viên xem các bảng giá dịch vụ đã khai báo.

Điều phối viên có thể xem thông tin khai báo nhân viên để truy cập chi tiết về các nhân viên; xem thông tin khai báo công nợ nhân viên để biết về các khoản công nợ của nhân viên; và xem thông tin khai báo ca làm việc để theo dõi lịch làm việc của các nhân viên và nhà cung cấp. Hệ thống cho phép điều phối viên xem thông tin khai báo hợp đồng thuê xe, truy cập chi tiết về các hợp đồng thuê xe; xem thông tin khai báo loại xe, để biết các loại xe đã được khai báo; và xem thông tin khai báo xe, truy cập chi tiết về từng xe. Ngoài ra, đối tác cung cấp xe có chức năng thêm, sửa, xóa thông tin xe của họ, cho phép quản lý thông tin xe của các đối tác cung cấp xe. [12]

Sơ đồ này thể hiện một hệ thống quản lý thông tin khai báo chi tiết, giúp điều phối viên có thể truy cập và xử lý một loạt các thông tin liên quan đến nhà cung cấp, nhân viên và xe một cách hiệu quả và dễ dàng.

#### Điều phối xe



Sơ đồ 3. 6 Mức dưới đỉnh - Điều phối xe

Sơ đồ DFD mức dưới đỉnh của chức năng điều phối xe mô tả chi tiết quy trình điều phối xe trong hệ thống, với điều phối viên và đối tác cung cấp xe có vai trò tương đương nhau. Cả hai đều chịu trách nhiệm điều phối xe, nhưng với sự phân chia rõ ràng về nhiệm vụ: điều phối viên quản lý và điều phối xe nội bộ, trong khi đối tác cung cấp xe điều phối xe của họ. Trung tâm của sơ đồ là hai tác nhân chính: Điều phối viên và Đối tác cung cấp xe. Cả hai đều có thể thực hiện các chức năng tương tự như nhau trong hệ thống.

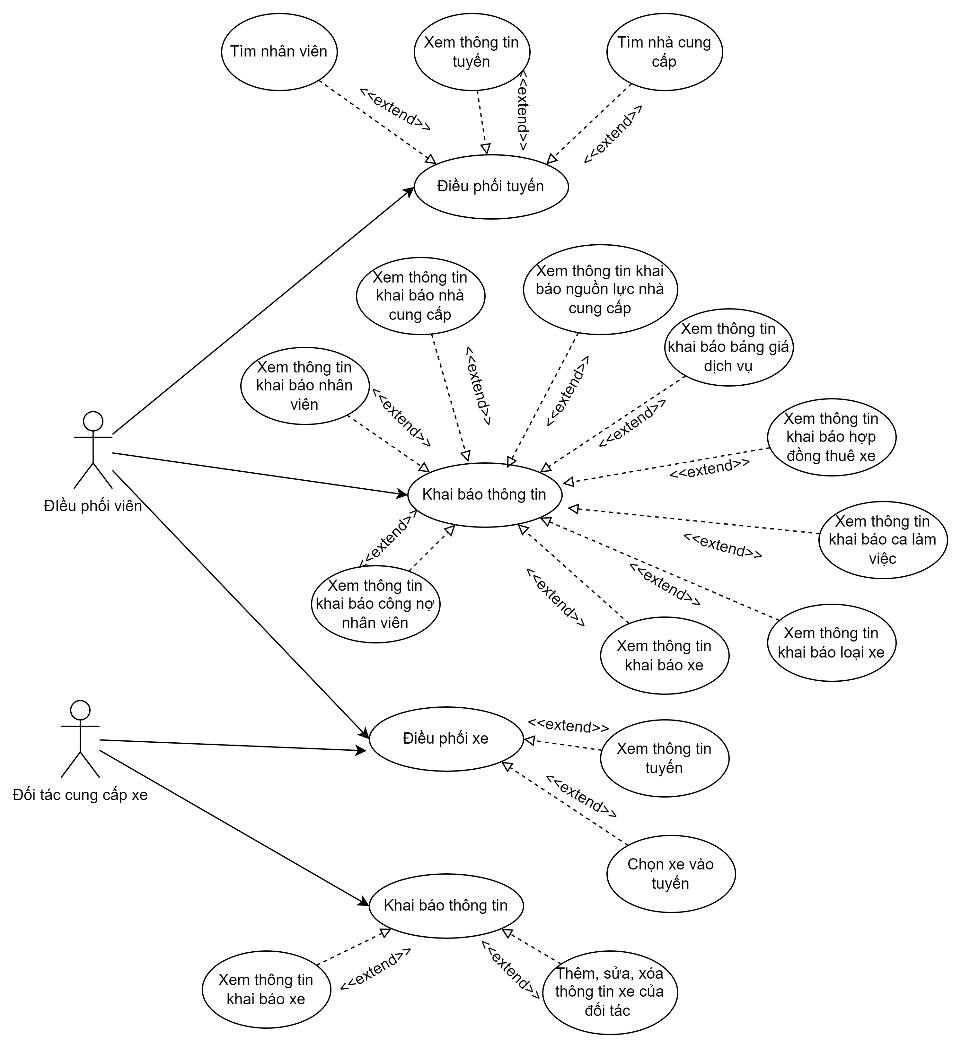
Với chức năng xem thông tin tuyến, cả điều phối viên và đối tác cung cấp xe có thể truy cập và xem thông tin chi tiết về các tuyến. Thông tin này bao gồm lộ trình, thời gian, điểm dừng, và các yêu cầu cụ thể của từng tuyến.

Với chức năng chọn xe vào tuyến, sau khi xem thông tin tuyến, cả điều phối viên và đối tác cung cấp xe sẽ chọn các xe phù hợp để phục vụ cho từng tuyến cụ thể. Việc này đảm bảo rằng xe được chọn đáp ứng các yêu cầu của tuyến và có khả năng thực hiện lộ trình một cách hiệu quả.

Điều phối viên và Đối tác cung cấp xe đều có thể yêu cầu thông tin về tuyến và hệ thống sẽ cung cấp dữ liệu tuyến thông qua “Xem thông tin tuyến”. Dựa vào thông tin tuyến, cả hai sẽ quyết định và chọn xe vào tuyến thông qua “Chọn xe vào tuyến”. Điều phối viên quản lý xe nội bộ trong khi Đối tác cung cấp xe quản lý và điều phối xe thuộc quyền sở hữu của họ. [12]

Sơ đồ này mô tả một quy trình điều phối xe linh hoạt và cân bằng, cho phép cả điều phối viên và đối tác cung cấp xe đều có thể quản lý và điều phối xe của mình một cách hiệu quả. Như vậy sẽ đảm bảo rằng tất cả các tuyến được phục vụ bởi các xe phù hợp, việc quản lý xe diễn ra trơn tru và nhất quán giữa các bên liên quan.

## **Sơ đồ use-case**



Sơ đồ 3. 7 Sơ đồ use case hệ thống

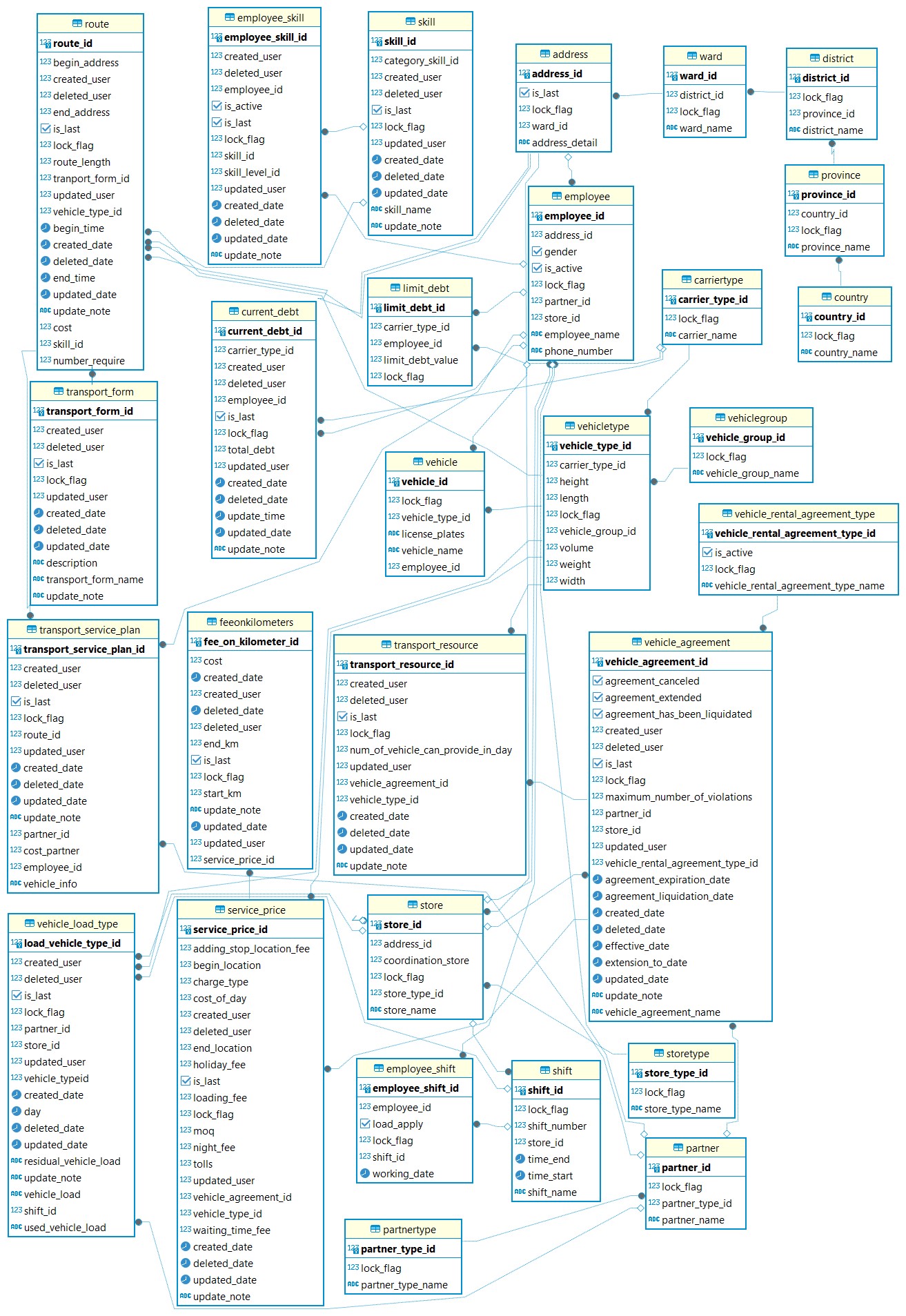
Sơ đồ use case mô tả các chức năng và quy trình tương tác giữa hai tác nhân chính là điều phối viên và đối tác cung cấp xe trong hệ thống. Cả hai tác nhân đều có các nhiệm vụ cụ thể nhưng cùng chung mục tiêu là đảm bảo việc điều phối và quản lý thông tin diễn ra hiệu quả và chính xác.

Điều phối viên chịu trách nhiệm quản lý và điều phối xe nội bộ. Các chức năng mà điều phối viên có thể thực hiện bao gồm: tìm nhân viên vào tuyến, cho phép điều phối viên chọn tìm thông tin về nhân viên trong hệ thống phù hợp với tuyến; xem thông tin tuyến, giúp điều phối viên truy cập và xem chi tiết các thông tin về tuyến, bao gồm lộ trình, thời gian, và các yêu cầu cụ thể của từng tuyến; tìm nhà cung cấp, cho phép điều phối viên chọn tìm thông tin về các nhà cung cấp xe để hệ thống tự động điều phối. Ngoài ra, điều phối viên còn có thể xem thông tin khai báo nhà cung cấp, thông tin khai báo nguồn lực nhà cung cấp, và thông tin khai báo bảng giá dịch vụ để đảm bảo nguồn lực và chi phí dịch vụ được quản lý chặt chẽ.

Điều phối viên cũng có quyền xem thông tin khai báo nhân viên để quản lý và theo dõi công việc, xem thông tin khai báo công nợ nhân viên và xem thông tin khai báo ca làm việc để quan sát công việc một cách hợp lý. Hơn nữa, điều phối viên có thể xem thông tin khai báo hợp đồng thuê xe, xem thông tin khai báo loại xe và thông tin khai báo xe để quản lý và điều phối xe nội bộ hiệu quả. Sau đó, điều phối viên chọn xe phù hợp vào tuyến để đảm bảo các lộ trình được thực hiện một cách hiệu quả.

Đối tác cung cấp xe chịu trách nhiệm điều phối và quản lý xe của họ. Các chức năng mà đối tác cung cấp xe có thể thực hiện bao gồm: xem thông tin tuyến, cho phép đối tác cung cấp xe truy cập và xem chi tiết các thông tin về tuyến để lên kế hoạch cung cấp xe phù hợp; chọn xe vào tuyến, giúp đối tác cung cấp xe chọn các xe của họ để phục vụ cho từng tuyến cụ thể, đảm bảo đáp ứng yêu cầu vận chuyển. Cuối cùng, đối tác cung cấp xe có thể thêm, sửa, xóa thông tin xe của mình để quản lý thông tin xe một cách chính xác. [13]

## **Mô hình ERD**



Sơ đồ 3. 8 Mô hình ERD

Mô hình ERD (Entity-Relationship Diagram) mô tả cấu trúc dữ liệu của hệ thống điều phối nguồn lực, bao gồm các thực thể (entities), các thuộc tính (attributes) và mối quan hệ (relationships) giữa chúng. Trong đó, thực thể route chứa thông tin về các tuyến như mã tuyến, địa chỉ bắt đầu, địa chỉ kết thúc, độ dài tuyến, và các thông tin liên quan khác. Thực thể này liên kết với các thực thể khác như transport\_form, employee, và skill để quản lý quy trình vận tải một cách hiệu quả.

Thực thể employee lưu trữ thông tin chi tiết về nhân viên, bao gồm mã nhân viên, địa chỉ, mã kho, số điện thoại, và các thuộc tính khác. Thực thể này liên kết với nhiều thực thể khác như address, ward, district, province, country, employee\_skill, employee\_shift, và vehicle để quản lý thông tin nhân lực trong hệ thống.

Thực thể vehicle chứa thông tin về các phương tiện vận tải, bao gồm mã phương tiện, loại phương tiện, biển số xe, tên phương tiện, và mã nhân viên lái xe. Thực thể này liên kết với các thực thể vehicle\_type, carrier\_type, vehicle\_group, vehicle\_rental\_agreement\_type, và store để quản lý và điều phối phương tiện một cách hiệu quả.

Thực thể store lưu trữ thông tin về các kho hàng, bao gồm mã kho, địa chỉ, loại kho, và tên kho. Thực thể này liên kết với address, store\_type, và transport\_service\_plan để quản lý thông tin kho hàng và các dịch vụ liên quan.

Thực thể transport\_service\_plan lưu trữ thông tin về kế hoạch dịch vụ vận chuyển, bao gồm mã kế hoạch, thông tin phương tiện, mã đối tác, và mã nhân viên. Thực thể này liên kết với các thực thể transport\_form, vehicle, employee, và partner để quản lý và điều phối dịch vụ vận chuyển.

Thực thể service\_price chứa thông tin về bảng giá dịch vụ, bao gồm mã giá dịch vụ, phí dừng thêm, địa điểm bắt đầu, chi phí theo ngày, và các thông tin khác. Thực thể này liên kết với vehicle\_type, fee\_on\_kilometers, và transport\_resource để quản lý giá cả dịch vụ vận tải.

Thực thể transport\_resource lưu trữ thông tin về nguồn lực nhà cung cấp, bao gồm mã nguồn lực, mã hợp đồng phương tiện, loại phương tiện, và giá dịch vụ. Thực thể này liên kết với các thực thể vehicle, vehicle\_agreement, và service\_price để quản lý tài nguyên vận tải một cách hiệu quả.

Thực thể vehicle\_agreement chứa thông tin về các hợp đồng phương tiện, bao gồm mã hợp đồng, mã kho, mã đối tác, loại hợp đồng thuê phương tiện, và các thông tin khác. Thực thể này liên kết với vehicle, store, partner, và vehicle\_rental\_agreement\_type để quản lý các hợp đồng phương tiện.

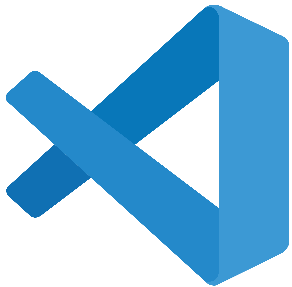
Ngoài ra, mô hình ERD còn bao gồm các thực thể khác như address, ward, district, province, country để lưu trữ thông tin địa chỉ và phân cấp địa lý; employee\_skill, skill để lưu trữ thông tin về kỹ năng của nhân viên; carrier\_type, vehicle\_type, vehicle\_group, vehicle\_rental\_agreement\_type để quản lý thông tin về loại phương tiện và các nhóm liên quan; partner, partner\_type để lưu trữ thông tin về đối tác và loại đối tác; employee\_shift, shift để quản lý thông tin về ca làm việc của nhân viên; current\_debt, limit\_debt để lưu trữ thông tin về nợ hiện tại và giới hạn nợ. [14]

Tổng quan, mô hình ERD đảm bảo rằng tất cả các thông tin cần thiết được quản lý một cách hợp lý và hiệu quả, từ đó hỗ trợ cho việc điều phối và quản lý vận tải một cách hiệu quả.

# CHƯƠNG 4. PHÁT TRIỂN ỨNG DỤNG

## **Công cụ phát triển**

### Visual Studio Code



Hình 4. 1 Logo Visual Studio Code

VS Code là một môi trường phát triển tích hợp (IDE) mã nguồn mở do Microsoft phát triển. Ra mắt lần đầu vào năm 2015, VS Code đã nhanh chóng trở thành một trong những IDE phổ biến nhất trên thế giới nhờ tính linh hoạt, khả năng mở rộng và hiệu suất cao. Giao diện của VS Code trực quan và dễ sử dụng, hoạt động trên nhiều hệ điều hành như Windows, macOS và Linux. Các thành phần chính của giao diện bao gồm thanh công cụ, trình biên tập mã, trình duyệt tệp và bảng điều khiển đầu ra, tất cả đều được thiết kế để nâng cao trải nghiệm người dùng.

VS Code hỗ trợ nhiều ngôn ngữ lập trình như Python, JavaScript, TypeScript, Java, C++, C#, PHP, Go và nhiều ngôn ngữ khác thông qua các tiện ích mở rộng, giúp lập trình viên có thể làm việc trên nhiều loại dự án khác nhau mà không cần phải chuyển đổi giữa các IDE. Khả năng hỗ trợ đa ngôn ngữ là một trong những lý do khiến VS Code được ưa chuộng rộng rãi.

Khả năng mở rộng của VS Code thông qua các tiện ích mở rộng là một trong những tính năng nổi bật nhất. Cửa hàng Extensions Marketplace cung cấp hàng ngàn tiện ích mở rộng giúp tăng cường chức năng của VS Code, từ hỗ trợ ngôn ngữ, tích hợp Git, đến các công cụ kiểm tra và gỡ lỗi. Người dùng có thể dễ dàng tìm kiếm và cài đặt các tiện ích này để tùy chỉnh môi trường phát triển theo nhu cầu cụ thể của mình.

Bên cạnh đó, VS Code còn hỗ trợ tích hợp nhiều công cụ phát triển khác nhau như Docker, Kubernetes, Azure và các dịch vụ đám mây khác, cho phép lập trình viên dễ dàng triển khai và quản lý ứng dụng trực tiếp từ VS Code mà không cần sử dụng thêm các công cụ bên ngoài. Tính năng tích hợp này tăng cường tính linh hoạt và tiện lợi của VS Code trong quá trình phát triển phần mềm.

Đặc biệt, VS Code là công cụ được dùng để phát triển toàn bộ dự án hiện tại, bao gồm cả backend, frontend và ứng dụng di động. Ngoài ra, VS Code có một cộng đồng lập trình viên đông đảo, cung cấp tài liệu, hướng dẫn và hỗ trợ phong phú. Microsoft cũng cung cấp tài liệu chính thức và các khóa học hướng dẫn chi tiết cho người dùng mới. [15]

### DBeaver



Hình 4. 2 Logo công cụ DBeaver

DBeaver là một công cụ quản lý cơ sở dữ liệu mã nguồn mở và miễn phí, được thiết kế để hỗ trợ nhiều hệ quản trị cơ sở dữ liệu khác nhau. Với giao diện thân thiện và dễ sử dụng, DBeaver cung cấp nhiều tính năng mạnh mẽ cho việc quản lý, thiết kế và tối ưu hóa cơ sở dữ liệu. Công cụ này có khả năng hỗ trợ đa nền tảng, chạy tốt trên các hệ điều hành phổ biến như Windows, macOS, và Linux. DBeaver nổi bật với khả năng kết nối với nhiều loại cơ sở dữ liệu khác nhau, từ các hệ quản trị phổ biến như MySQL, PostgreSQL, Oracle, SQL Server, đến các hệ quản trị ít phổ biến hơn như SQLite, Firebird, và Apache Derby. Ngoài ra, DBeaver còn hỗ trợ kết nối với các cơ sở dữ liệu đám mây và cung cấp các tính năng nâng cao như tích hợp với hệ thống kiểm soát phiên bản, hỗ trợ scripting và các công cụ phân tích hiệu năng cơ sở dữ liệu.

Trong đề tài hiện tại, PostgreSQL được sử dụng làm hệ quản trị cơ sở dữ liệu chính và DBeaver làm công cụ quản lý. Khi sử dụng DBeaver để quản lý PostgreSQL, ta có thể tận dụng các tính năng của DBeaver để thực hiện nhiều tác vụ quản lý cơ sở dữ liệu một cách hiệu quả.

DBeaver cung cấp giao diện đồ họa trực quan giúp dễ dàng thực hiện các thao tác như tạo và quản lý các bảng, thực hiện các truy vấn SQL và quản lý quyền truy cập của người dùng. Công cụ này còn hỗ trợ việc nhập xuất dữ liệu, sao lưu và phục hồi cơ sở dữ liệu một cách đơn giản và nhanh chóng. Chúng khá quan trọng đối với đề tài, nơi yêu cầu việc quản lý dữ liệu một cách chính xác và hiệu quả.

Ngoài ra còn khả năng hiển thị và chỉnh sửa cấu trúc bảng, lập chỉ mục và thiết lập các ràng buộc dữ liệu một cách dễ dàng thông qua giao diện đồ họa. Như vậy giúp tiết kiệm thời gian và công sức trong việc thiết kế và tối ưu hóa cơ sở dữ liệu. [16]

### Draw.io



Hình 4. 3 Logo công cụ draw.io

Draw.io là một công cụ vẽ sơ đồ trực tuyến miễn phí, cho phép người dùng tạo ra các biểu đồ, sơ đồ và bản vẽ kỹ thuật một cách dễ dàng và nhanh chóng. Với giao diện người dùng thân thiện và trực quan, draw.io hỗ trợ nhiều loại sơ đồ khác nhau như sơ đồ quy trình (flowchart), sơ đồ mạng (network diagram), sơ đồ tổ chức (org chart), và nhiều loại sơ đồ UML (Unified Modeling Language). Công cụ này có thể được sử dụng trực tiếp trên trình duyệt web hoặc thông qua các phiên bản cài đặt trên máy tính. draw.io tích hợp tốt với các nền tảng đám mây như Google Drive, OneDrive và Dropbox, giúp người dùng dễ dàng lưu trữ, chia sẻ và cộng tác trên các bản vẽ của mình. Bên cạnh đó, draw.io còn cung cấp nhiều mẫu sơ đồ và biểu tượng phong phú, hỗ trợ người dùng trong việc thiết kế và trình bày các ý tưởng một cách rõ ràng và hiệu quả.

Trong đề tài hiện tại, draw.io được sử dụng để phân tích và thiết kế hệ thống, nhằm tạo ra các sơ đồ minh họa chi tiết về cấu trúc và hoạt động của hệ thống. Draw.io cung cấp các công cụ cần thiết để tạo ra các sơ đồ thiết kế hệ thống một cách hiệu quả và chính xác. Với Draw.io, có thể dễ dàng vẽ các sơ đồ quy trình để biểu diễn các bước thực hiện trong hệ thống, từ đó giúp ta hiểu rõ hơn về quy trình hoạt động và các mối quan hệ giữa các thành phần trong hệ thống.

Một ưu điểm lớn của Draw.io là khả năng cộng tác trực tuyến, cho phép ta dễ dàng chia sẻ các sơ đồ thiết kế với các bên liên quan và nhận phản hồi kịp thời, giúp đảm bảo rằng các ý tưởng và thiết kế được kiểm tra và cải tiến liên tục dựa trên sự góp ý của đồng nghiệp và khách hàng. [17]

## **Phát triển back-end**

### Cơ sở dữ liệu

#### PostgreSQL



Hình 4. 4 Logo PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở rất mạnh mẽ và được sử dụng rộng rãi trên toàn thế giới. Được phát triển từ hơn 30 năm trước, PostgreSQL đã khẳng định được vị thế của mình như là một trong những lựa chọn hàng đầu trong lĩnh vực quản trị cơ sở dữ liệu.

Một trong những điểm nổi bật của PostgreSQL là tính mã nguồn mở, cho phép người dùng có thể sử dụng, thay đổi và phân phối lại phần mềm một cách tự do. Chúng mang lại lợi ích lớn cho các tổ chức và cá nhân muốn tối ưu hóa chi phí và độ linh động trong việc phát triển ứng dụng.

PostgreSQL hỗ trợ đa nền tảng, có thể chạy trên nhiều hệ điều hành như Linux, Windows, macOS và BSD, giúp cho người dùng có thể triển khai trên môi trường họ đã quen thuộc mà không gặp phải các rào cản kỹ thuật.

Hệ quản trị cơ sở dữ liệu này cũng nổi bật với tính năng tuân thủ chuẩn SQL nghiêm ngặt và cung cấp các tính năng nâng cao như ACID (Atomicity, Consistency, Isolation, Durability), đảm bảo tính nhất quán và độ tin cậy của dữ liệu đối với các ứng dụng yêu cầu sự bảo đảm về vấn đề đó.

PostgreSQL có khả năng mở rộng từ các ứng dụng nhỏ đến các hệ thống phức tạp thông qua các tính năng như replication, partitioning và clustering làm cho nó trở thành lựa chọn phù hợp cho các dự án phát triển lớn có yêu cầu cao về khả năng mở rộng và hiệu suất.

Đối với đề tài đang sử dụng PostgreSQL để lưu trữ dữ liệu của hệ thống, việc sử dụng các bảng được thiết kế từ mô hình ERD và các quan hệ rõ ràng là vô cùng quan trọng. PostgreSQL cung cấp một cơ sở dữ liệu quan hệ mạnh mẽ và đầy đủ các tính năng để đáp ứng các yêu cầu này một cách hiệu quả. [18]

#### Model hệ thống

Hệ thống triển khai mô hình Database First với Hibernate để kết nối và tương tác với cơ sở dữ liệu PostgreSQL. Hibernate là một framework ORM (Object-Relational Mapping) mạnh mẽ cho Java, giúp ánh xạ dữ liệu từ cơ sở dữ liệu quan hệ sang các đối tượng trong lập trình Java một cách tự động và dễ dàng. Việc sử dụng Hibernate trong mô hình Database First cho phép đội phát triển xây dựng cấu trúc dữ liệu trong cơ sở dữ liệu PostgreSQL trước, sau đó sử dụng các công cụ và chức năng của Hibernate để tự động tạo ra các đối tượng Java tương ứng (entities) và các lớp DAO (Data Access Objects).

Mô hình này giúp cho quá trình phát triển trở nên hiệu quả hơn bằng cách tận dụng sức mạnh của Hibernate trong việc quản lý các thao tác CRUD (Create, Read, Update, Delete) với cơ sở dữ liệu. Hibernate tự động tạo và quản lý các câu truy vấn SQL phù hợp với PostgreSQL, giảm thiểu công việc lập trình viên phải thủ công viết và duy trì các câu truy vấn SQL. [19]

Sau đây là chuỗi kết nối tới postgreSQL qua hibernate:

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres

spring.datasource.username=postgres

spring.datasource.password=1234

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.show-sql=false

spring.jpa.open-in-view=false

#remove null values responed

spring.jackson.default-property-inclusion = non\_null

**spring.datasource.url**: Đây là URL kết nối đến cơ sở dữ liệu PostgreSQL. Trong ví dụ này, PostgreSQL đang chạy trên localhost (127.0.0.1) trên cổng 5432 và tên cơ sở dữ liệu là postgres.

**spring.datasource.username** và **spring.datasource.password**: Lần lượt là tên người dùng và mật khẩu để đăng nhập vào cơ sở dữ liệu PostgreSQL.

**spring.jpa.hibernate.ddl-auto**: Cấu hình này xác định cách Hibernate sẽ quản lý các bảng trong cơ sở dữ liệu. Trong trường hợp này, giá trị none cho biết Hibernate sẽ không thực hiện bất kỳ thay đổi nào vào cấu trúc cơ sở dữ liệu.

**spring.jpa.properties.hibernate.dialect**: Đây là dialect của Hibernate dành cho PostgreSQL, giúp Hibernate biết cách tương tác với cơ sở dữ liệu PostgreSQL.

**spring.jpa.properties.hibernate.format\_sql**: Cấu hình này cho phép Hibernate định dạng các câu lệnh SQL được hiển thị trong log để dễ đọc hơn và dễ dàng debug.

**spring.jpa.show-sql**: Cấu hình này quy định liệu Hibernate có nên hiển thị các câu lệnh SQL được tạo ra trong log hay không. Trong ví dụ này, giá trị là false, tức là Hibernate sẽ không hiển thị các câu lệnh SQL trong log.

**spring.jpa.open-in-view**: Cấu hình này cho biết liệu Spring Boot có nên mở rộng Session Hibernate cho mọi request hay không. Giá trị false cho biết rằng không nên mở rộng, giúp tránh các vấn đề về hiệu suất trong ứng dụng.

**spring.jackson.default-property-inclusion**: Cấu hình Jackson để xác định cách xử lý thuộc tính mặc định khi chuyển đổi đối tượng Java thành JSON. Giá trị non\_null chỉ định rằng chỉ những thuộc tính không phải là null mới được bao gồm trong đầu ra JSON.

Các cấu hình này cùng nhau tạo nên một môi trường ổn định và hiệu quả để ứng dụng Spring Boot kết nối và tương tác với cơ sở dữ liệu PostgreSQL thông qua Hibernate một cách chính xác và hiệu quả.

Sau khi đã thiết lập kết nối thành công với cơ sở dữ liệu PostgreSQL, hệ thống sử dụng mô hình Database First để tự động tạo các đối tượng Java (model) tương ứng với các bảng đã được định nghĩa trong cơ sở dữ liệu. Ví dụ, để minh họa cho mô hình “Country”, ta sẽ có các thuộc tính và quan hệ như sau:

@Entity(name=“COUNTRY”)

public class Country implements Serializable {

@Id

@Column(unique=true, nullable=false, precision=10)

private BigDecimal countryId;

private String countryName;

@OneToMany(mappedBy=“country”)

@JsonIgnore

private Set<Province> province;

public Country() {

super();

}

Trong mô hình Country được minh họa:

**@Entity(name = “COUNTRY”)**: Đánh dấu lớp này là một Entity trong Hibernate, tương ứng với bảng COUNTRY trong cơ sở dữ liệu.

**@Id**: Xác định trường countryId là khóa chính của bảng COUNTRY.

**@Column(unique = true, nullable = false, precision = 10)**: Cấu hình cho trường countryId trong cơ sở dữ liệu. Được đánh dấu là unique (duy nhất), không cho phép null và có độ chính xác là 10.

**private String countryName**: Thuộc tính này tương ứng với cột countryName trong bảng COUNTRY, lưu trữ tên của quốc gia.

**@OneToMany(mappedBy = “country”)**: Đánh dấu mối quan hệ một-nhiều với mô hình Province thông qua trường country trong Province. Mỗi quốc gia có thể có nhiều tỉnh/thành phố (Province).

**@JsonIgnore**: Chỉ định rằng khi chuyển đổi đối tượng Country thành định dạng JSON, thuộc tính provinces sẽ được bỏ qua để tránh vòng lặp đệ quy.

### Công nghệ Spring Boot



Hình 4. 5 Logo Spring Boot

Spring Boot là một phần mở rộng của framework Spring, được thiết kế để giúp phát triển các ứng dụng Java độc lập một cách dễ dàng và nhanh chóng. Được phát triển bởi Pivotal Team, Spring Boot tối ưu hóa quy trình phát triển bằng cách cung cấp cấu hình tự động, các công cụ mạnh mẽ, và nhiều tính năng hữu ích khác.

Một trong những điểm nổi bật của Spring Boot là cấu hình tự động. Thay vì phải cấu hình thủ công từng thành phần và thư viện, Spring Boot tự động phát hiện và cấu hình dựa trên các phụ thuộc đã khai báo trong dự án, giúp giảm thiểu thời gian và công sức của nhà phát triển, cho phép họ tập trung vào xây dựng logic nghiệp vụ thay vì phải xử lý các chi tiết cấu hình phức tạp.

Spring Boot cũng cho phép xây dựng các ứng dụng Java có thể chạy độc lập mà không cần một máy chủ ứng dụng riêng biệt nhờ tích hợp các máy chủ nhúng như Tomcat, Jetty hoặc Undertow vào trong ứng dụng. Kết quả là các ứng dụng Spring Boot có thể được triển khai và chạy như các ứng dụng Java thông thường mà không cần cấu hình bổ sung cho máy chủ.

Hỗ trợ mạnh mẽ cho phát triển các dịch vụ RESTful là một lợi thế lớn của Spring Boot. Các nhà phát triển có thể dễ dàng tạo các lớp điều khiển (controller) để xử lý các yêu cầu HTTP và trả về các phản hồi ở định dạng JSON hoặc XML. Spring Boot đơn giản hóa việc xây dựng API RESTful, giúp các nhà phát triển nhanh chóng thiết kế và triển khai các dịch vụ web hiện đại. [20]

Sau đây là các bước chi tiết để xây dựng hoàn chỉnh một API lấy danh sách tuyến (route) bằng cách đi qua các lớp repository, service, và controller.

#### Repository

Đầu tiên, ta cần tạo một lớp Repository để tương tác với cơ sở dữ liệu. Repository trong Spring Data JPA là một giao diện mở rộng từ JpaRepository, cung cấp nhiều phương thức tiện ích để thao tác với cơ sở dữ liệu như lưu, xóa, và tìm kiếm các thực thể. Trong trường hợp này, ta định nghĩa một Repository cho thực thể Route như sau:

public interface IRouteRepository extends JpaRepository<Route, BigDecimal> {

@Query(value = “select \* from route order by route\_id”, nativeQuery = true)

public List<Route> getAllRoute();

}

Khai báo bắt đầu với việc định nghĩa IRouteRepository và kế thừa từ JpaRepository với Route là thực thể được quản lý và BigDecimal là kiểu dữ liệu của khóa chính (route\_id). Việc kế thừa từ JpaRepository cung cấp một tập hợp các phương thức CRUD (Create, Read, Update, Delete) cơ bản và mạnh mẽ, giúp ta không cần phải viết lại các phương thức như save(), findById(), findAll(), deleteById(), v.v.

Annotation @Repository đánh dấu giao diện này là một Repository trong Spring, cho phép Spring quản lý nó như một thành phần trong context, giúp các thành phần khác của ứng dụng có thể dễ dàng thêm phụ thuộc (dependency injection) vào và sử dụng.

Phương thức truy vấn tùy chỉnh được định nghĩa bằng cách sử dụng @Query. Trong trường hợp này, ta sử dụng câu truy vấn SQL gốc để lấy dữ liệu từ bảng route và sắp xếp theo route\_id. Việc sử dụng nativeQuery = true cho phép sử dụng SQL gốc thay vì JPQL (Java Persistence Query Language), cung cấp hiệu suất tốt hơn trong một số tình huống và cho phép truy cập trực tiếp vào các đặc điểm cụ thể của cơ sở dữ liệu.

Phương thức getAllRoute() trả về một danh sách các đối tượng Route được lấy từ cơ sở dữ liệu, sắp xếp theo route\_id, đảm bảo rằng dữ liệu được lấy ra theo một thứ tự xác định, giúp việc xử lý dữ liệu ở các lớp khác của ứng dụng trở nên dễ dàng và trực quan hơn.

#### Service

Service trong Spring là một lớp chịu trách nhiệm xử lý logic nghiệp vụ và giao tiếp với lớp Repository để truy vấn dữ liệu. Trong trường hợp này, ta sẽ tạo một lớp Service triển khai giao diện IRouteService và sử dụng lớp Repository để lấy danh sách các tuyến từ cơ sở dữ liệu.

@EnableCaching

@Service

public class RouteServiceImpl implements IRouteService {

@Autowired

private final IRouteRepository routeRepository;

@Override

public List<Route> getAllRoute() {

return routeRepository.getAllRoute();

}

}

Annotation @EnableCaching được sử dụng để kích hoạt khả năng caching trong Spring. Caching giúp cải thiện hiệu suất của ứng dụng bằng cách lưu trữ kết quả của các truy vấn dữ liệu để sử dụng lại trong tương lai, thay vì phải thực hiện lại truy vấn mỗi lần có yêu cầu mới.

Khai báo lớp RouteServiceImpl triển khai IRouteService. Lớp ảo này định nghĩa các phương thức mà lớp Service phải triển khai, đảm bảo tính nhất quán và rõ ràng trong việc xử lý nghiệp vụ liên quan đến các tuyến.

Triển khai phương thức getAllRoute() được định nghĩa trong IRouteService. Phương thức này gọi phương thức getAllRoute() của IRouteRepository để lấy danh sách các tuyến từ cơ sở dữ liệu và trả về kết quả.

Service này đóng vai trò quan trọng trong kiến trúc của ứng dụng, tách biệt rõ ràng giữa logic nghiệp vụ và truy vấn dữ liệu, nó không chỉ giúp mã nguồn dễ dàng bảo trì và mở rộng mà còn tăng tính linh hoạt khi cần thay đổi hoặc mở rộng logic nghiệp vụ trong tương lai.

#### Controller

Controller trong Spring là một lớp chịu trách nhiệm xử lý các yêu cầu HTTP từ client, gọi lớp Service để lấy dữ liệu và trả kết quả về cho client dưới dạng JSON hoặc các định dạng khác. Trong trường hợp này, ta sẽ tạo một lớp Controller để xử lý các yêu cầu liên quan đến các tuyến.

@RestController

@RequestMapping(“/api/route”)

@CrossOrigin(“\*”)

public class RouteController {

@Autowired

private final IRouteService routeService;

@GetMapping(““)

public ResponseEntity<List<Route>> getAllRoute() {

List<Route> routes = routeService.getAllRoute();

return new ResponseEntity<>(routes, HttpStatus.OK);

}

}

Annotation @RestController đánh dấu lớp này là một controller trong Spring, cho phép Spring quản lý nó như một thành phần trong context. Nó cũng cho phép các phương thức trong lớp này trả về dữ liệu trực tiếp dưới dạng JSON hoặc các định dạng khác, thay vì trả về một view.

Annotation @RequestMapping(“/api/route”) định nghĩa URL gốc cho tất cả các phương thức trong lớp này. Mọi yêu cầu đến /api/route sẽ được xử lý bởi các phương thức trong lớp này.

Annotation @CrossOrigin(“\*”) cho phép tất cả các nguồn (origins) có thể truy cập vào các tài nguyên của controller này. Chúng rất hữu ích trong việc phát triển ứng dụng web, đặc biệt khi frontend và backend chạy trên các domain khác nhau.

Phương thức getAllRoute() được định nghĩa với @GetMapping(“”), chỉ định rằng phương thức này sẽ xử lý các yêu cầu GET đến URL /api/route. Phương thức này gọi getAllRoute() của IRouteService để lấy danh sách các tuyến và trả về kết quả dưới dạng ResponseEntity với trạng thái HTTP là OK.

Controller này đóng vai trò quan trọng trong kiến trúc của ứng dụng, làm cầu nối giữa client và các lớp dịch vụ. Nó xử lý các yêu cầu HTTP từ client, gọi lớp Service để lấy dữ liệu và trả kết quả về cho client.

Sau khi đã có API, dữ liệu của tuyến được trả về khi gọi như sau:

[ {

"routeId": 1,

"beginTime": "2024-04-01T08:00:00",

"endTime": "2024-04-01T10:00:00",

"routeLength": 14.00,

"vehicleType": {

"vehicleTypeId": 3,

"length": 3.50,

"weight": 2400.00,

"width": 1.65,

"height": 1.70,

"volume": 9.82,

"vehiclegroup": {

"vehicleGroupId": 1,

"vehicleGroupName": "xe tải thường"

}

},

"cost": 3000000.00,

"skill": {

"skillId": 1,

"skillName": "Giao hàng",

},

"address": {

"addressId": 1,

"addressDetail": "Cụm 2-4, đường M14, Khu công nghiệp Tân Bình Mở Rộng",

"ward": {

"wardId": 1,

"wardName": "Phường Tân Bình",

"district": {

"districtId": 13,

"districtName": "Quận Bình Tân",

"province": {

"provinceId": 1,

"provinceName": "Hồ Chí Minh",

"country": {

"countryId": 1,

"countryName": "Việt Nam"

}

}

}

}

}

"transportForm": {

"transportFormId": 1,

"transportFormName": "Vận chuyển thường",

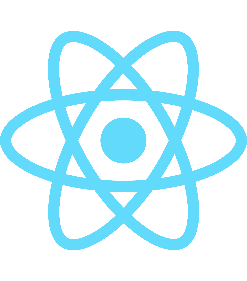
}

}]

## **Phát triển front-end website**

Website sử dụng công nghệ Reactjs và tập trung chính vào người dùng là điều phối viên, với mục tiêu hỗ trợ họ quản lý và điều phối tuyến của kho một cách hiệu quả và chính xác. Hệ thống sẽ được thiết kế để tự động hoá các quy trình điều phối tìm nhà cung cấp và nhân viên, đồng thời cung cấp các khai báo và thông tin chi tiết cần thiết. Nhờ đó, điều phối viên có thể dễ dàng quan sát, nắm bắt thông tin và thực hiện tác vụ điều phối xe vào tuyến đã được gán cho công ty nội bộ một cách nhanh chóng và chính xác. Cụ thể, giao diện sẽ hiển thị một cách rõ ràng và trực quan tất cả các thông tin liên quan đến tuyến, nhà cung cấp và nhân viên.

### Công nghệ ReactJS



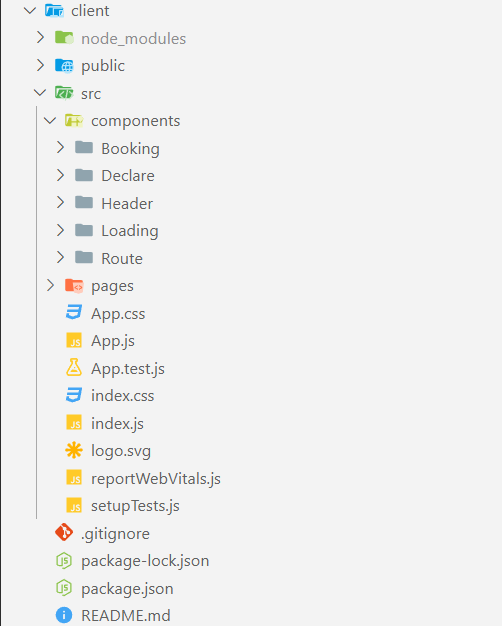
Hình 4. 6 Logo ReactJS

ReactJS là một thư viện JavaScript mạnh mẽ và phổ biến được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. Được phát triển bởi Facebook và ra mắt lần đầu tiên vào năm 2013, ReactJS đã nhanh chóng trở thành một trong những công cụ phát triển front-end ưa chuộng nhất trong cộng đồng lập trình. Với khả năng tạo ra các UI phức tạp và hiệu suất cao, ReactJS giúp các nhà phát triển dễ dàng xây dựng và quản lý các ứng dụng web hiện đại.

Một trong những đặc điểm nổi bật của ReactJS là kiến trúc dựa trên thành phần (Component-Based Architecture). Với kiến trúc này, UI được chia thành các thành phần nhỏ, độc lập và có thể tái sử dụng. Mỗi thành phần bao gồm cả logic và giao diện của một phần cụ thể của ứng dụng, giúp việc quản lý mã nguồn trở nên dễ dàng và hiệu quả hơn. Bằng cách sử dụng các thành phần, lập trình viên có thể xây dựng các phần của ứng dụng một cách độc lập và sau đó kết hợp chúng lại để tạo ra một UI phức tạp hơn.

ReactJS sử dụng một cơ chế gọi là Virtual DOM (DOM ảo) để cải thiện hiệu suất. Khi trạng thái của ứng dụng thay đổi, Virtual DOM sẽ tính toán sự khác biệt giữa DOM hiện tại và DOM mới, sau đó chỉ cập nhật những phần thay đổi trên DOM thực, giúp tối ưu hóa việc render và giảm thiểu tác động đến hiệu suất của ứng dụng. Bằng cách này, ReactJS đảm bảo rằng chỉ những phần thực sự cần cập nhật mới được thay đổi, giúp ứng dụng chạy mượt mà và nhanh chóng. [21]

Sau đây là tổ chức các thư mục trong ReactJS của hệ thống:



Hình 4. 7 Tổ chức ReactJS của hệ thống

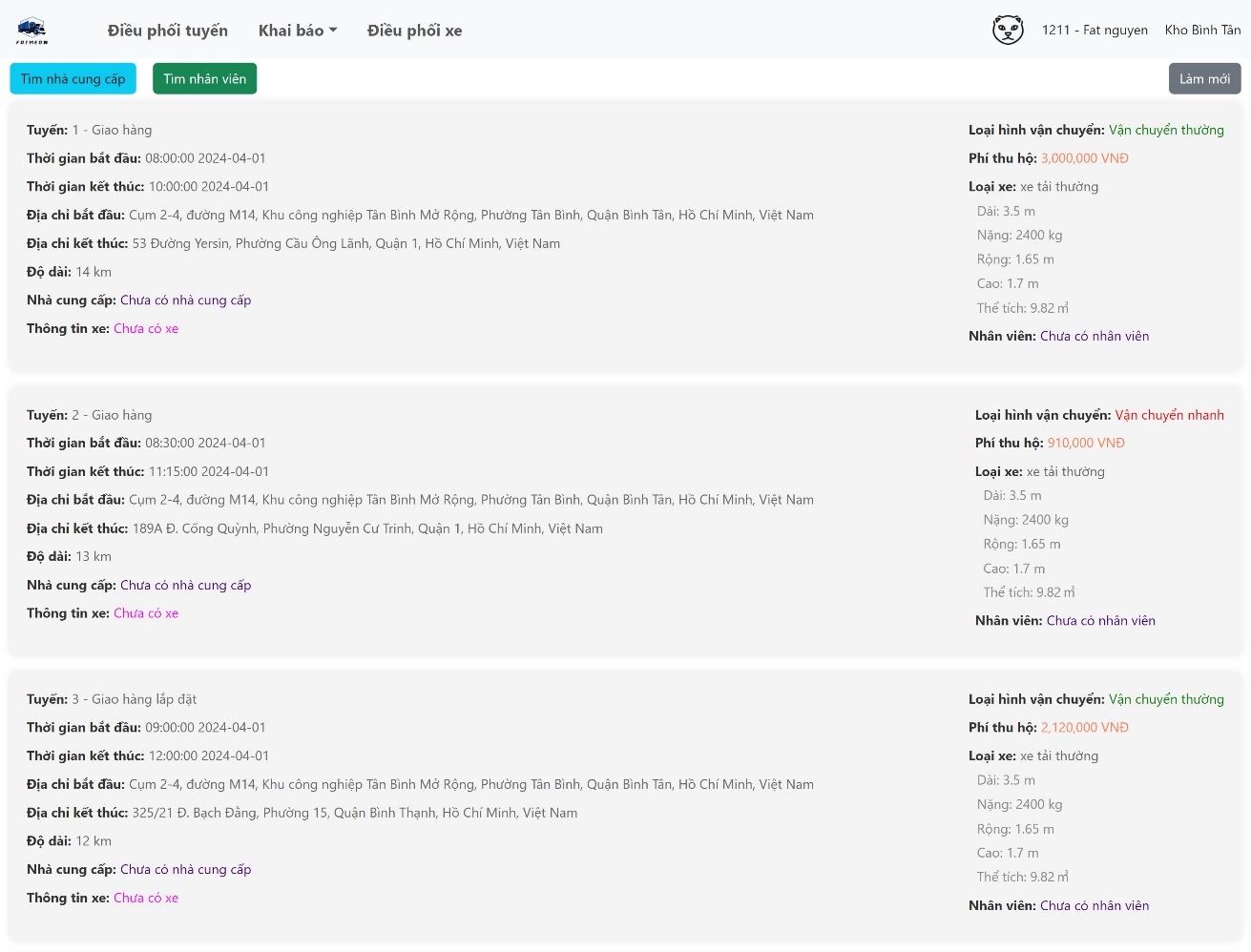
Thư mục gốc của dự án là client/, bên trong đó có các thư mục con quan trọng như node\_modules/, public/, và src/. Thư mục node\_modules/ chứa tất cả các gói (packages) npm được cài đặt cho dự án, đây là nơi lưu trữ các thư viện và công cụ hỗ trợ quá trình phát triển ứng dụng. Thư mục public/ chứa các tệp tĩnh có thể truy cập công khai, thường bao gồm các tệp như index.html, các hình ảnh, favicon, và các tài nguyên khác không cần phải xử lý bởi React.

Thư mục chính chứa mã nguồn của ứng dụng là src/. Bên trong src/, có nhiều thư mục con và tệp tin khác nhau để tổ chức mã nguồn một cách logic và có cấu trúc. Thư mục components/ chứa các thành phần React được sử dụng trong ứng dụng. Các thành phần này được tổ chức thành các thư mục con như Booking/, Declare/, Header/, Loading/, và Route/, mỗi thư mục con này chứa các thành phần liên quan đến các tính năng cụ thể của website như điều phối xe, khai báo, header, trạng thái tải và điều phối tuyến.

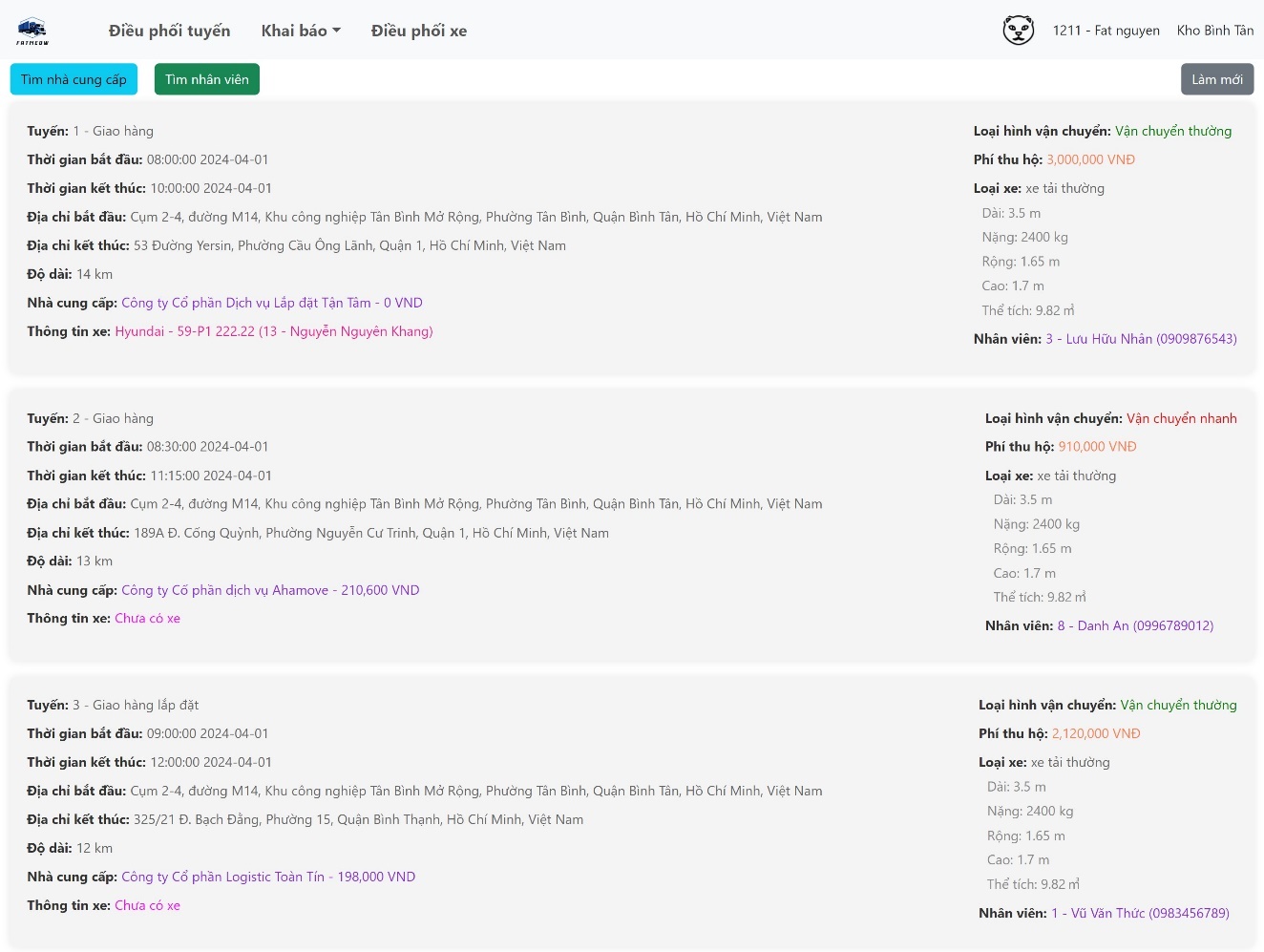
Thư mục pages/ chứa các trang của ứng dụng, mỗi trang là một thành phần lớn bao gồm nhiều thành phần nhỏ hơn từ thư mục components. Các tệp tin khác trong src/ bao gồm App.css, App.js, App.test.js, index.css, index.js, logo.svg, reportWebVitals.js, và setupTests.js. Trong đó, App.css chứa các định dạng (styles) cho thành phần App, App.js chứa mã nguồn chính của thành phần App nơi cấu trúc chính của ứng dụng được định nghĩa.

Ngoài ra, trong thư mục gốc còn có các tệp package.json, package-lock.json, .gitignore, và README.md. Tệp .gitignore chứa cấu hình để git bỏ qua các tệp và thư mục không cần theo dõi. Tệp package-lock.json tự động được tạo ra bởi npm để khóa phiên bản các gói đã cài đặt, đảm bảo tính nhất quán khi cài đặt lại các gói. Tệp package.json chứa thông tin về dự án và danh sách các gói phụ thuộc (dependencies). Và tệp README.md chứa thông tin mô tả về dự án, cách cài đặt và sử dụng. Các phần tiếp theo là chi tiết giao diện website của hệ thống bao gồm điều phối tuyến, những thông tin khai báo và điều phối xe.

### Giao diện điều phối tuyến



Hình 4. 8 Màn hình điều phối tuyến chưa gán thông tin



Hình 4. 9 Màn hình điều phối tuyến đã gán thông tin

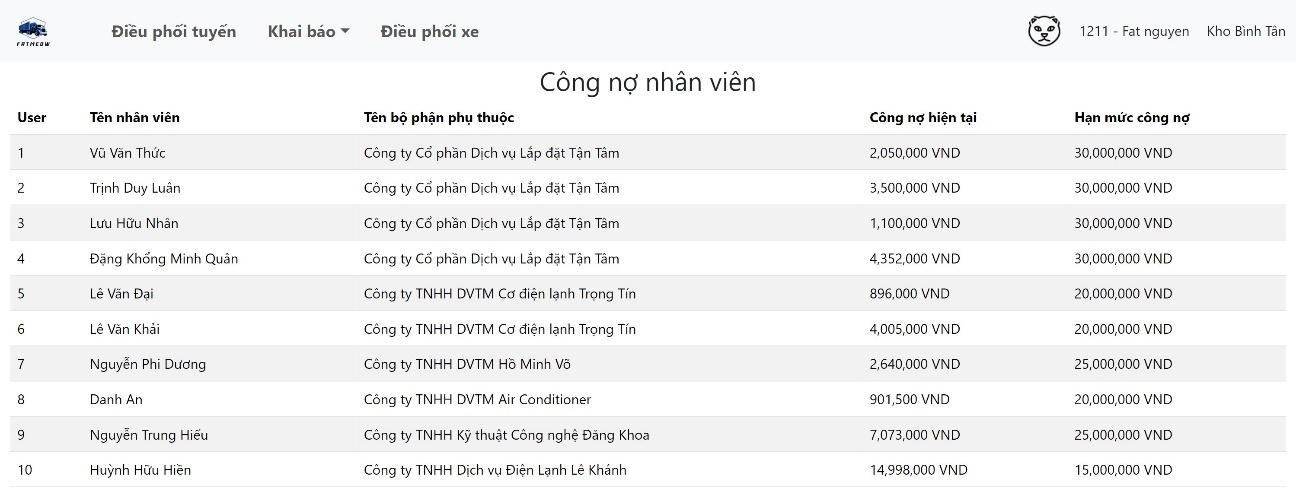
### Giao diện thông tin khai báo



Hình 4. 10 Màn hình thông tin khai báo Bảng giá dịch vụ của các đối tác

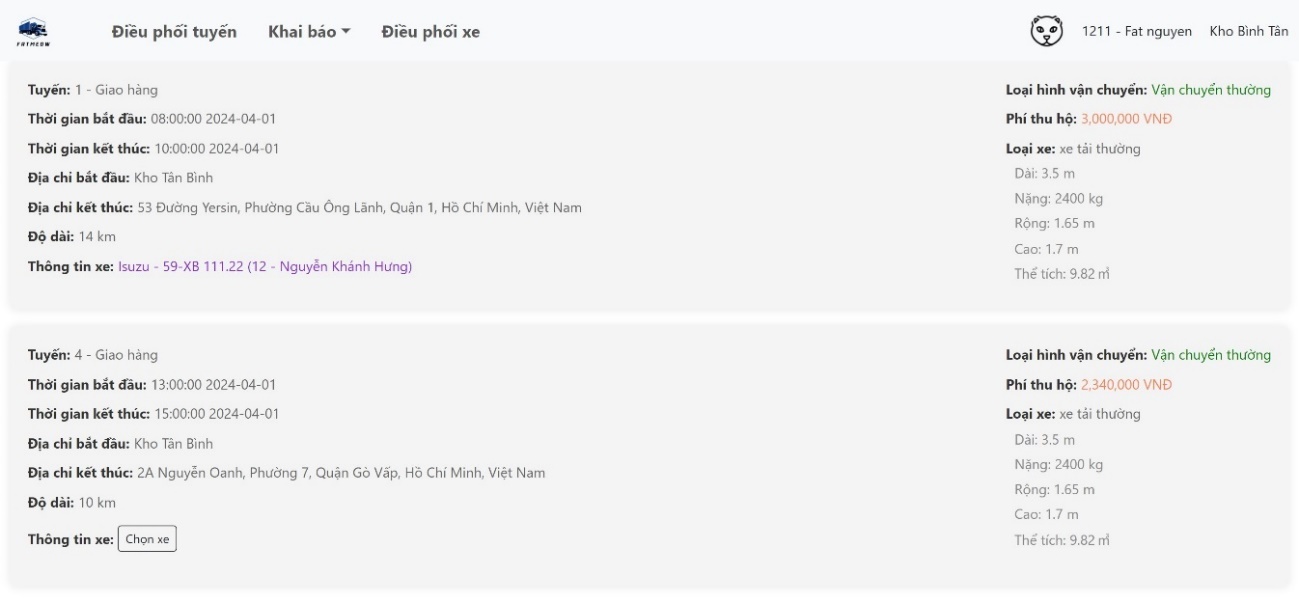


Hình 4. 11 Màn hình thông tin khai báo Nhân viên



Hình 4. 12 Màn hình khai báo Công nợ nhân viên

### Giao diện điều phối xe



Hình 4. 13 Màn hình điều phối xe vào tuyến

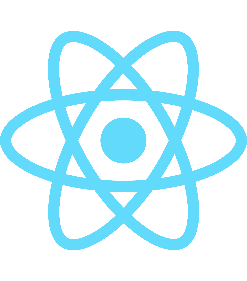
## **Phát triển front-end ứng dụng di động**

Ứng dụng di động sử dụng công nghệ React Native và sẽ tập trung vào việc hỗ trợ nhân viên của các công ty vận tải nhỏ lẻ, những doanh nghiệp chưa có hệ thống quản lý chuyên nghiệp. Mục tiêu chính của hệ thống là giúp các công ty này quản lý đội xe tải một cách hiệu quả và nhận phản hồi từ khách hàng về các tuyến cần gán xe nhanh chóng, giúp họ xử lý kịp thời và gán xe chính xác.

Hệ thống sẽ bao gồm một trang quản lý xe, nơi người dùng có thể thêm, sửa, hoặc xóa thông tin về các xe tải của công ty, giúp quản lý dữ liệu xe một cách dễ dàng, bảo đảm thông tin luôn được cập nhật chính xác và đầy đủ.

Ngoài ra, ứng dụng còn có một trang điều phối xe, nơi nhân viên có thể chọn xe tải và tài xế phù hợp để gán vào các tuyến đang chờ xử lý. Trang điều phối này cho phép người dùng xem nhanh danh sách các tuyến cần gán xe, cùng với các thông tin chi tiết về từng tuyến như thời gian khởi hành, điểm xuất phát và điểm đến, yêu cầu tuyến và các điều kiện cần thiết khác.

### Công nghệ React Native



Hình 4. 14 Logo React Native

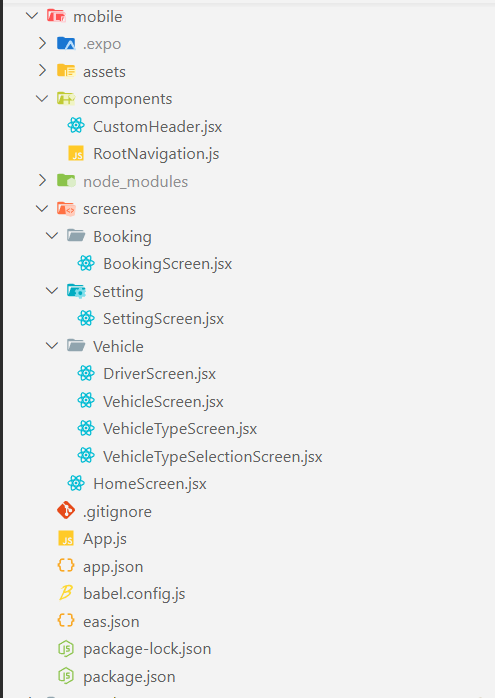
React Native là một framework được phát triển bởi Facebook, cho phép lập trình viên xây dựng ứng dụng di động đa nền tảng bằng JavaScript và React. Có nghĩa là bạn có thể sử dụng cùng một mã nguồn để phát triển ứng dụng cho cả iOS và Android, giảm thiểu thời gian và chi phí phát triển so với việc phát triển song song hai ứng dụng riêng biệt.

Một trong những lợi ích nổi bật của React Native là sử dụng lại các thành phần giao diện người dùng (UI components) đã có sẵn từ React và áp dụng chúng vào trong ứng dụng di động. Như vậy sẽ giúp cho việc phát triển nhanh chóng hơn và dễ dàng hơn, đồng thời giảm thiểu lỗi phát sinh trong quá trình phát triển.

Với React Native, lập trình viên có thể sử dụng JavaScript để điều khiển luồng logic và hiển thị giao diện người dùng, trong khi các phần phức tạp hơn như tích hợp với các thư viện và module bên thứ ba cũng có sẵn thông qua việc sử dụng các bản mô phỏng (wrapper) hoặc các thư viện Native Modules.

Một điểm mạnh khác của React Native là khả năng tối ưu hóa hiệu suất. Thay vì sử dụng WebView như các ứng dụng hybrid, React Native sử dụng các thành phần native để tạo ra giao diện và tương tác, mang lại trải nghiệm người dùng mượt mà và gần giống với ứng dụng native. [22]

Sau đây là tổ chức các thư mục trong React Native của hệ thống:



Hình 4. 15 Tổ chức React Native của hệ thống

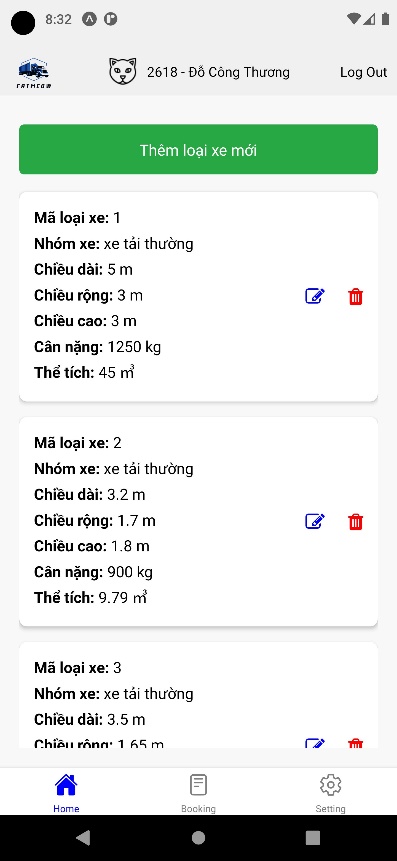
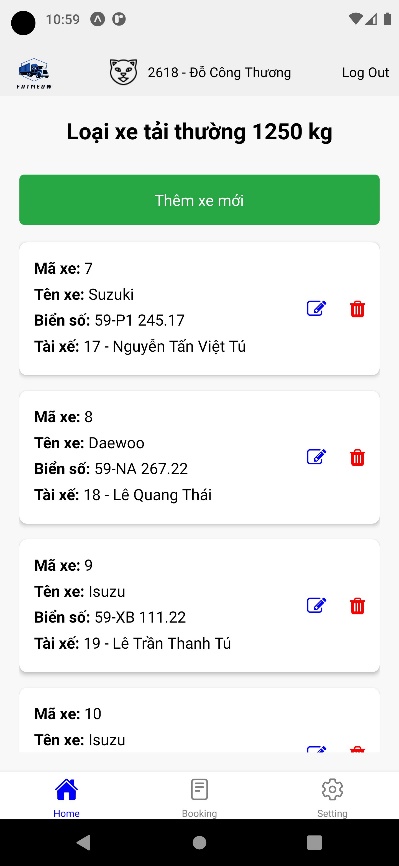
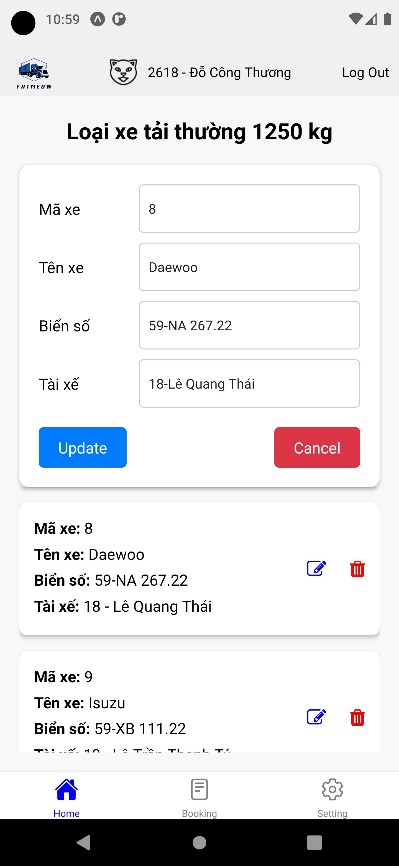
Thư mục gốc của dự án là mobile/, bên trong đó có các thư mục con quan trọng như node\_modules/, assets/, components/, screens/. Thư mục node\_modules/ chứa tất cả các gói (packages) npm được cài đặt cho dự án. Thư mục assets/ chứa các tệp tĩnh có thể truy cập công khai, thường bao gồm các hình ảnh, favicon, và các tài nguyên khác không cần phải xử lý.

Thư mục components/ chứa các thành phần React được sử dụng trong ứng dụng. Các thành phần này bao gồm CustomHeader là header của ứng dụng và RootNavigation để cấu hình các liên kết màn hình trong ứng dụng.

Thư mục screens/ chứa các màn hình chính của ứng dụng, là một thư mục lớn bao gồm nhiều thư mục nhỏ hơn. Cụ thể thư mục Booking/ là màn hình để điều phối xe, thư mục Setting/ hiển thị màn hình cài đặt và thư mục Vehicle/ chứa các màn hình quản lý khai báo các thông tin liên quan đến loại xe, xe, tài xế. Các tệp tin khác bao gồm App.js, app.json, babel.config.js và eas.json. Trong đó, App.js chứa mã nguồn chính của thành phần App nơi cấu trúc chính của ứng dụng được định nghĩa.

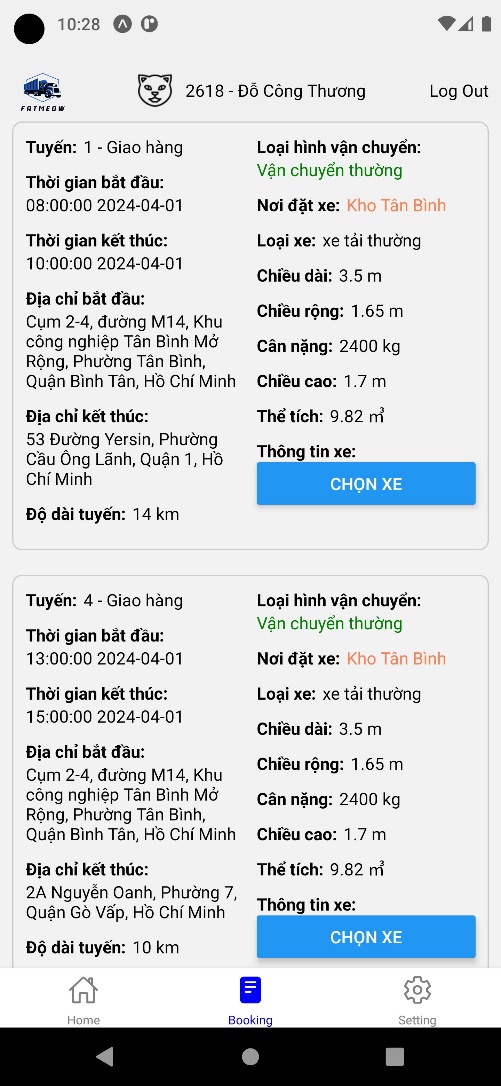
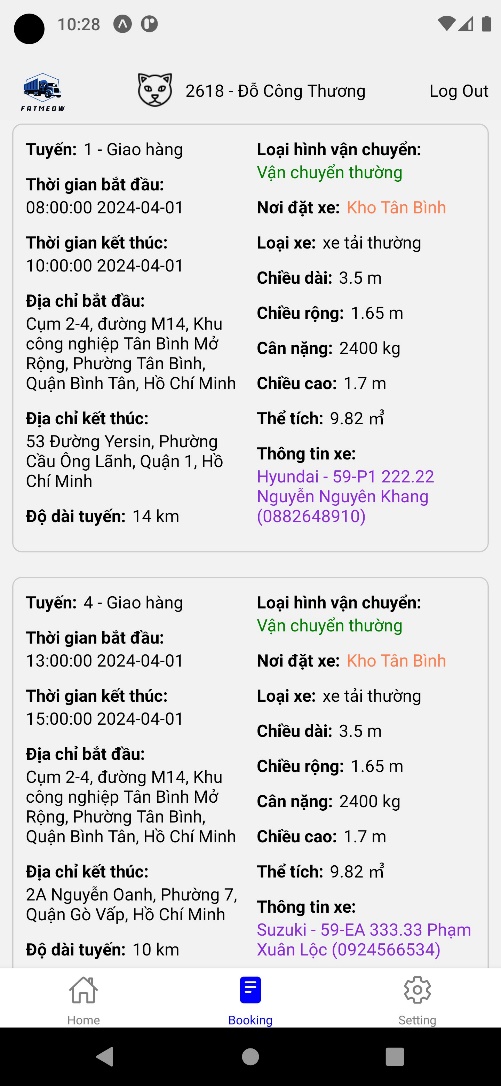
Các phần tiếp theo là chi tiết giao diện ứng dụng dị động của hệ thống bao gồm quản lý thông tin xe và điều phối xe.

### Quản lý thông tin xe (Loại xe và xe)

Hình 4. 16 Màn hình quản lý loại xe và xe

### Điều phối xe

Hình 4. 17 Màn hình điều phối xe của đối tác

# KẾT QUẢ VÀ KIẾN NGHỊ

## **Kết quả đạt được**

Trong quá trình thực hiện đề tài “Nghiên cứu và ứng dụng công cụ OR-Tools vào xây dựng hệ thống điều phối nguồn lực tự động”, em đã đạt được những kết quả như sau. Đầu tiên, phát triển hoàn chỉnh hệ thống điều phối tự động, sử dụng OR-Tools kết hợp với các công nghệ hiện đại như Java Spring Boot, ReactJS, React Native và PostgreSQL. Hệ thống này đã tối ưu hóa việc quản lý và phân bổ nguồn lực trong lĩnh vực logistics, từ đó nâng cao hiệu suất hoạt động và giảm thiểu chi phí cho doanh nghiệp.

Nhờ vào hệ thống điều phối tự động, chất lượng dịch vụ của các doanh nghiệp logistics sẽ được nâng cao đáng kể. Hệ thống giúp giảm thiểu thời gian và công sức trong việc điều phối nguồn lực, từ đó cải thiện sự hài lòng của khách hàng. Như vậy có ý nghĩa rất lớn đối với các doanh nghiệp trong việc duy trì và nâng cao chất lượng dịch vụ của mình.

## **Hạn chế**

Hệ thống vẫn còn tồn tại một số hạn chế nhất định. Một trong những hạn chế là phụ thuộc vào chất lượng dữ liệu đầu vào. Nếu dữ liệu đầu vào không chính xác hoặc không đầy đủ, hệ thống sẽ khó đưa ra các giải pháp tối ưu. Thêm vào đó, hệ thống chủ yếu tập trung vào các bài toán tuyến tính, do đó chưa thể tối ưu hoàn toàn cho các bài toán phi tuyến hoặc phi tuyến tính tổng quát. Vấn đề này có thể hạn chế khả năng áp dụng của hệ thống vào các tình huống thực tế phức tạp hơn.

## **Kiến nghị**

Để tiếp tục phát triển và cải thiện hệ thống, em đưa ra một số hướng đi mới và các cải tiến mạnh mẽ hơn. Đầu tiên, cần nâng cao khả năng xử lý của hệ thống bằng cách cải thiện thuật toán và tối ưu hóa mô hình, khi áp dụng các kỹ thuật tiên tiến như học máy (machine learning) để dự đoán và tối ưu hóa quy trình điều phối sẽ giúp hệ thống xử lý hiệu quả hơn các bài toán lớn và phức tạp.

Tiếp theo, cần nghiên cứu và ứng dụng thêm các công cụ và kỹ thuật khác để mở rộng phạm vi ứng dụng của hệ thống, không chỉ dừng lại ở lĩnh vực logistics mà còn có thể được áp dụng vào các lĩnh vực khác, bao gồm cả các bài toán phi tuyến tính. Như vậy sẽ mở ra nhiều cơ hội mới cho việc ứng dụng công nghệ vào quản lý và điều phối nguồn lực.

Cải thiện giao diện người dùng cũng là một phần quan trọng để tăng cường trải nghiệm trong quá trình sử dụng hệ thống. Nâng cấp các tính năng hiển thị thông tin và tương tác trực quan hơn với người dùng, giúp hệ thống trở nên thân thiện và dễ sử dụng hơn.

Những kiến nghị này nhằm mục tiêu không chỉ cải thiện hiệu suất của hệ thống mà còn mở rộng khả năng ứng dụng và nâng cao chất lượng dịch vụ cho các doanh nghiệp logistics, góp phần vào sự phát triển bền vững và hiệu quả của ngành.

# TÀI LIỆU THAM KHẢO

[6]. *Giáo trình Nhập môn trí tuệ nhân tạo*, PGS.TS Từ Minh Phương, NXB Thông tin và Truyền thông - 2017.

[12]. *Giáo trình Phân tích và thiết kế hệ thống thông tin*, TS. Cao Thị Nhạn, Nguyễn Đình Loan Phương, Đại Học Quốc Gia HCM - 2019.

[13]. *Giáo trình Phân tích thiết kế hướng đối tượng*, Trương Ninh Thuận, Đặng Đức Hạnh, Đại học Quốc gia Hà Nội - 2016.

[14]. *Giáo trình Cơ sở dữ liệu*, Nguyễn Đăng Tỵ, Đỗ Phúc, Đại Học Quốc Gia HCM - 2010.

[1]. *Hands-On Guide to Google OR-Tools*, Ravindra K. Ahuja, James B. Orlin.

[7]. *A Multipurpose Backtracking Algorithm*, Hilary Priestley, Martin Ward.

[15]. *Visual Studio Code in Introductory Computer Science Course: An Experience Report*, Jialiang Tan, Yu Chen, Shuyin Jiao, 2023.

[16]. *Introduction to DBeaver for data exploration*, Squarespace, 2021.

[18]. *PostgreSQL: Introduction and Concepts*, Bruce Momjian, 2000.

[19]. *Java Persistence with Hibernate*, CHRISTIAN BAUER, GAVIN KING, GARY GREGORY, 2016.

[20]. *Spring Boot*, tutorialspoint, 2018.

[21]. *ReactJS*, tutorialspoint, 2021.

[22]. *React Native for Mobile Development*, Akshat Paul, Abhishek Nalwaya, 2019.

[2]. *Advanced LP Solving,* <https://developers.google.com/optimization/lp/lp_advanced>, truy cập lúc 20/05/2024.

[3]. *Constraint Optimization*, https://developers.google.com/optimization/cp, truy cập lúc 20/05/2024.

[4]. *SCIP,* <https://www.scipopt.org/doc/html/index.php>, truy cập lúc 20/05/2024.

[5]. *GLPK: The Gnu Linear Programming Kit,* <https://docs.racket-lang.org/glpk/index.html>, truy cập lúc 20/05/2024.

[8]. *Propagation Algorithm*, <https://www.sciencedirect.com/topics/computer-science/propagation-algorithm>, truy cập ngày 21/05/2024.

[9]. *Introduction to Branch and Bound – Data Structures and Algorithms Tutorial*, <https://www.geeksforgeeks.org/introduction-to-branch-and-bound-data-structures-and-algorithms-tutorial/>, truy cập ngày 21/05/2024.

[10]. *Solving an Assignment Problem,* <https://developers.google.com/optimization/assignment/assignment_example>, truy cập ngày 21/05/2024.

[11]. *Assignment with Teams of Workers,* <https://developers.google.com/optimization/assignment/assignment_teams>, truy cập ngày 22/05/2024.

[17]. *Draw.io online – a step-by-step guide for users*, <https://fotc.com/blog/draw-io-online-guide/>, truy cập ngày 23/05/2024.