**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PBL2: DỰ ÁN CƠ SỞ LẬP TRÌNH**

**Đề tài : QUẢN LÝ SINH VIÊN KTX**

**SINH VIÊN THỰC HIỆN:**

**TRẦN NHẬT NGUYÊN LỚP: 23T\_DT1 NHÓM: 23.Nh10B**

**TRẦN VIẾT TÂM LỚP: 23T\_Nhat2 NHÓM: 23.Nh10B**

**GIẢNG VIÊN HƯỚNG DẪN: TS. Đặng Thiên Bình**

**Đà Nẵng 30/11/2024**

LỜI MỞ ĐẦU

Ký túc xá là nơi ở và học tập của hàng ngàn sinh viên từ khắp mọi miền Tổ quốc. Với số lượng sinh viên ngày càng gia tang, việc quản lý ký túc xá trở thành một nhiệm vụ phức tạp và đòi hỏi sự chính xác, nhanh chóng, cũng như là hiệu quả trong công tác vận hành. Những phương pháp quản lý truyền thống dựa trên giấy tờ hoặc bảng tính thủ công dần bộc lội nhiều hạn chế như mất mát dữ liệu, khó truy xuất thông tin, và mất nhiều thời gian xử lý.

Trong bối cảnh đó, việc ứng dụng thành tựu công nghệ thông tin vào thực tiễn quản lý Ký túc xá không chỉ là xu hướng mà còn là nhu cầu tất yếu. Một hệ thống quản lý ký túc xá hiện đại, tích hợp các chức năng thêm sinh viên, lưu trữ dữ liệu sinh viên vào cơ sở dữ liệu, quản lý phòng ở, theo dõi các khoản thu chi, hỗ trợ tra cứu nhanh chóng, và đồng thời tạo dữ liệu phân tích thống kê nhằm trực quan hóa dữ liệu sẽ giúp nâng cao chất lượng quản lý, tiết kiệm thời gian, cũng như đảm bảo tính minh bạch, chính xác, và ngoài ra còn giúp xây dựng một hệ thống thông minh, có tính ứng dụng cao trong thực tiễn.

Đề tài “Quản lý sinh viên Ký túc xá” này được thực hiện cho hệ thống Ký túc xá thành phố Đà Nẵng, nhằm xây dựng một giải pháp công nghệ hiệu quả, thông minh, đáp ứng nhu cầu quản lý thực tiễn tại Ký túc xá này. Hệ thống sẽ tập trung vào các chức năng cơ bản nhưng thiết yếu, đồng thời có khả năng mở rộng và tích hợp các giải pháp thông minh trong bối cảnh thời đại công nghệ số đang phát triển mạnh. Hệ thống xây dựng trên giao diện đồ họa dòng lệnh (TUI – Terminal User Interface), trên nguyên tắc đơn giản, gọn nhẹ, nhưng vẫn đảm bảo độ đẹp mắt và tính ổn định tương đối cao.

Hy vọng đề tài này không chỉ góp phần giải quyết bài toán quản lý Ký túc xá mà còn là tiền đề để ứng dụng công nghệ thông tin và quá trình phân tích dữ liệu vào các lĩnh vực quản lý khác, đóng góp vào sự phát triển chung của toàn xã hội, góp phần hoàn thành sự nghiệp công nghiệp hóa – hiện đại hóa của nước nhà, hoàn thiện mục tiêu tiến lên Xã hội chủ nghĩa của Việt Nam nói riêng và của toàn nhân loại nói chung.

Thân ái.

MỤC LỤC

[1](#_heading=h.1ksv4uv)

[LỜI MỞ ĐẦU 2](#_heading=h.30j0zll)

[MỤC LỤC 3](#_heading=h.1fob9te)

[DANH MỤC HÌNH VẼ 4](#_heading=h.3znysh7)

[1.](#_heading=h.2et92p0) GIỚI THIỆU ĐỀ TÀI 5

[2.](#_heading=h.44sinio) PHÂN TÍCH CHỨC NĂNG HỆ THỐNG 5

[2.1.](#_heading=h.2jxsxqh)  5

[2.2.](#_heading=h.z337ya)  5

[2.3.](#_heading=h.3j2qqm3)  5

[3.](#_heading=h.1y810tw) THIẾT KẾ CẤU TRÚC DỮ LIỆU 5

[3.1.](#_heading=h.tyjcwt) Phát biểu bài toán 5

[3.2.](#_heading=h.4i7ojhp) Phân tích và ứng dụng cấu trúc dữ liệu trong hệ thống 5

[4.](#_heading=h.2xcytpi) PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG VÀ TRIỂN KHAI HỆ THỐNG 5

[4.1.](#_heading=h.1ci93xb) Cấu trúc hệ thống hướng đối tượng (class, object, relation) 5

[4.2.](#_heading=h.3dy6vkm) Kết quả 5

[4.2.1.](#_heading=h.1t3h5sf) Giao diện chính của chương trình 5

[4.2.2.](#_heading=h.4d34og8) Kết quả thực thi của chương trình 5

[4.2.3.](#_heading=h.2s8eyo1) Nhận xét 5

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 5](#_heading=h.17dp8vu)

[a.](#_heading=h.3rdcrjn) Kết luận 5

[b.](#_heading=h.3rdcrjn) Hướng phát triển 5

[TÀI LIỆU THAM KHẢO 6](#_heading=h.26in1rg)

[[1] Tên tác giả, Tên tài liệu, Tên nhà xuất bản, năm xuất bản 6](#_heading=h.lnxbz9)

[[2] Tên chủ sở hữu, Tên bài viết, url, ngày truy cập 6](#_heading=h.35nkun2)

DANH MỤC HÌNH VẼ

[*Hình 2.1: Màn hình welcome* 8](#_Toc185385244)

[*Hình 2.2: Màn hình menu chính* 9](#_Toc185385245)

[*Hình 2.3: Phần giới thiệu của tính năng Thêm sinh viên* 10](#_Toc185385246)

[*Hình 2.4: Giới thiệu của tính năng Đăng ký bao phòng* 10](#_Toc185385247)

[*Hình 2.5: Giới thiệu của tính năng Danh sách Sinh viên* 11](#_Toc185385248)

[*Hình 2.6: Giới thiệu của tính năng Danh sách Phòng* 11](#_Toc185385249)

[*Hình 2.7: Giới thiệu của tính năng In tài liệu* 11](#_Toc185385250)

[*Hình 2.8: Giới thiệu của tính năng In bản đồ* 12](#_Toc185385251)

[*Hình 2.9: Giới thiệu của tính năng Thống Kê* 12](#_Toc185385252)

[*Hình 2.10: Chức năng thêm sinh viên* 13](#_Toc185385253)

[*Hình 2.11: Lỗi nếu người dùng nhập thiếu thông tin* 13](#_Toc185385254)

[*Hình 2.12: Tính năng chọn phòng cho sinh viên* 14](#_Toc185385255)

[*Hình 2.13: Chọn phòng cho sinh viên* 15](#_Toc185385256)

# GIỚI THIỆU ĐỀ TÀI

Đề tài “Quản lý sinh viên KTX” được triển khai với mục tiêu xây dựng một hệ thống hỗ trợ quản lý sinh viên lưu trú hiện đại, thông minh, chính xác và tối ưu đối với Ký túc xá phía Tây thành phố Đà Nẵng, tại địa chỉ 08 Hà Văn Tính, Hòa Khánh Nam, Liên Chiểu, Đà Nẵng hiện tại, và dối với các hệ thống ký túc xá khác trên toàn quốc trong tương lai không xa. Hệ thống này được thiết kế nhằm:

* Thêm, xóa, sửa sinh viên một cách dễ dàng với sự ổn định cao.
* Quản lý danh sách sinh viên lưu trú, thông tin của sinh viên đó một cách chi tiết và rõ ràng.
* Đăng ký bao phòng hỗ trợ sinh viên có thể đăng ký chung phòng với người quen.
* Quản lý danh sách phòng hiệu quả, kịp thời thông báo khi phòng hỏng và bảo trì.
* Hỗ trợ tra cứu thông tin sinh viên theo tên sinh viên bao gồm tìm kiếm theo thời gian thực (Realtime-search) và tìm kiếm nhanh (sử dụng Binary search), tra cứu thông tin phòng bằng tìm kiếm thời gian thực.
* Xuất ra bản đồ Ký túc xá (file jpg), hỗ trợ tân sinh viên dễ dàng tìm kiếm, hiểu rõ sơ đồ cấu trúc của Ký túc xá.
* Xuất ra danh sách sinh viên, danh sách phòng (file csv), hỗ trợ in ấn, kiểm kê, phân tích dữ liệu sinh viên bằng các công cụ bên ngoài.
* Thống kê số lượng sinh viên, số lượng phòng, dữ liệu phân tích (số lượng sinh viên nữ/nam/khác, phòng bảo trì/có thể sử dụng, phòng đủ/chưa đủ sinh viên).

Ngoài ra, hệ thống còn được xây dựng và tích hợp những nền tảng công nghệ hiệu quả, tối ưu nhằm phát huy tối đa hiệu quả của công nghệ:

* Hệ quản trị cơ sở dữ liệu LevelDB, là hệ quản trị dựa trên Non-SQL, có tốc độ xử lý và độ bảo mật cao dựa trên cơ chế Level nhưng chưa được sử dụng nhiều, hiện nay LevelDB đang áp dụng trong một số ít những ứng dụng phổ biến như AutoCAD Mechanical, AutoCAD Inventor, …
* Giao diện đồ họa trên dòng lệnh (TUI – Terminal User Interface), hiện đại, đơn giản, gọn nhẹ nhưng vẫn đẹp mắt, mặc dù còn nhiều giới hạn và hạn chế về hiển thị so với GUI. TUI hiện tại đang áp dụng trong một số ứng dụng phổ biến như neofetch, htop, vim, neovim, …
* Kiến trúc phần mềm MVVM (Model – View – Viewmodel), là mẫu kiến trúc phổ biến trong phát triển ứng dụng, đặc biệt là ứng dụng giao diện người dung, tận dụng tối đa ưu điểm dễ dàng kiểm thử, xây dựng ứng dụng chuyên nghiệp.
* Lập trình hướng đối tượng (OOP – Object-oriented programming) và Design Pattern (Factory, Singleton, Adapter, Prototyping, …) hỗ trợ xây dựng ứng dụng hiệu quả và dễ bảo trì.
* Các cấu trúc dữ liệu và thuật toán là yếu tố quan trọng góp phần xây dựng ứng dụng “Quản lý sinh viên KTX” hiệu quả, tối ưu và dễ dàng.

Ứng dụng không chỉ giảm tải khối lượng công việc quản lý mà còn tăng tính chính xác, minh bạch và hiệu quả vận hành. Hệ thống được phát triển với giao diện đồ họa dòng lệnh thân thiện, đẹp mắt, dễ sử dụng trên nền tảng ngôn ngữ lập trình C++, hệ thống tạo trình biên dịch đa nền tảng CMake, và có khả năng tích hợp, mở rộng để đáp ứng nhu cầu quản lý Ký túc xá có quy mô khác nhau, đồng thời có thể chạy trên hầu hết những hệ điều hành phổ biến nhất hiện nay như Windows, macOs, các hệ điều hành dựa trên Linux,…

Để dễ dàng quản lý và phát triển ứng dụng thì nhóm đã sử dụng các công cụ như Git, Trello, Google Meet, Messenger, … hỗ trợ hiệu quả hơn trong quá trình giao tiếp và xây dựng ứng dụng. Ngoài ra, nhóm định hướng phát triển phần mềm theo một mô hình phát triển dựa trên mô hình phát triển phần mềm linh hoạt (Incremental Development), nhằm tiếp cận người dùng là nhân viên ký túc xá trong thực trạng ký túc xá ngày càng mở rộng và tăng số lượng dịch vụ, đồng thời giúp nhóm giảm bớt áp lực thời gian cũng như là về mặt giới hạn kiền thức.

Với sự phát triển ngày càng mạnh mẽ của công nghệ thông tin, đặc biệt là khối ngành về dữ liệu (như là khoa học dữ liệu – Data Science, trí tuệ nhân tạo – Artificial Intelligence), việc ứng dụng công nghệ cũng như là các cơ chế quản lý, phân tích và trực quan hóa dữ liệu vào quản lý ký túc xá không chỉ mang lại lợi ích cho đội ngũ quản lý mà còn tạo điều kiện thuận lợi cho sinh viên trong quá trình học tập và sinh hoạt. Đầy chính là động lực để nhóm thực hiện đề tài này, đóng góp vào việc nâng cao chất lượng quản lý và dịch vụ tại các ký túc xá, đồng thời giúp nhóm có thêm nguồn kinh nghiệm và kiến thức quý giá trong quá trình tìm hiểu, học tập và xây dung cũng như là tối ưu hóa hệ thống, là hành trang quan trọng giúp cho nhóm có thể có nền tảng tốt để phát triển nhiều hơn các ứng dụng góp phần vào quá trình phát triển và tiến lên xã hội chủ nghĩa của Việt Nam.

# PHÂN TÍCH CHỨC NĂNG HỆ THỐNG

A diagram of a company

Description automatically generated

*Hình 2.1: Tổng quan sơ lược chức năng hệ thống*

Trong phần này, nhóm không chỉ giới thiệu về chức năng hệ thống, giải thích cấu trúc của hình trên, kết hợp với kết quả thực thi của chương trình để có cái nhìn rõ nét về hoạt động và cấu trúc hệ thống của phần mềm quản lý ký túc xá.

## Màn hình welcome

A screenshot of a computer

Description automatically generated

*Hình 2.2: Màn hình welcome*

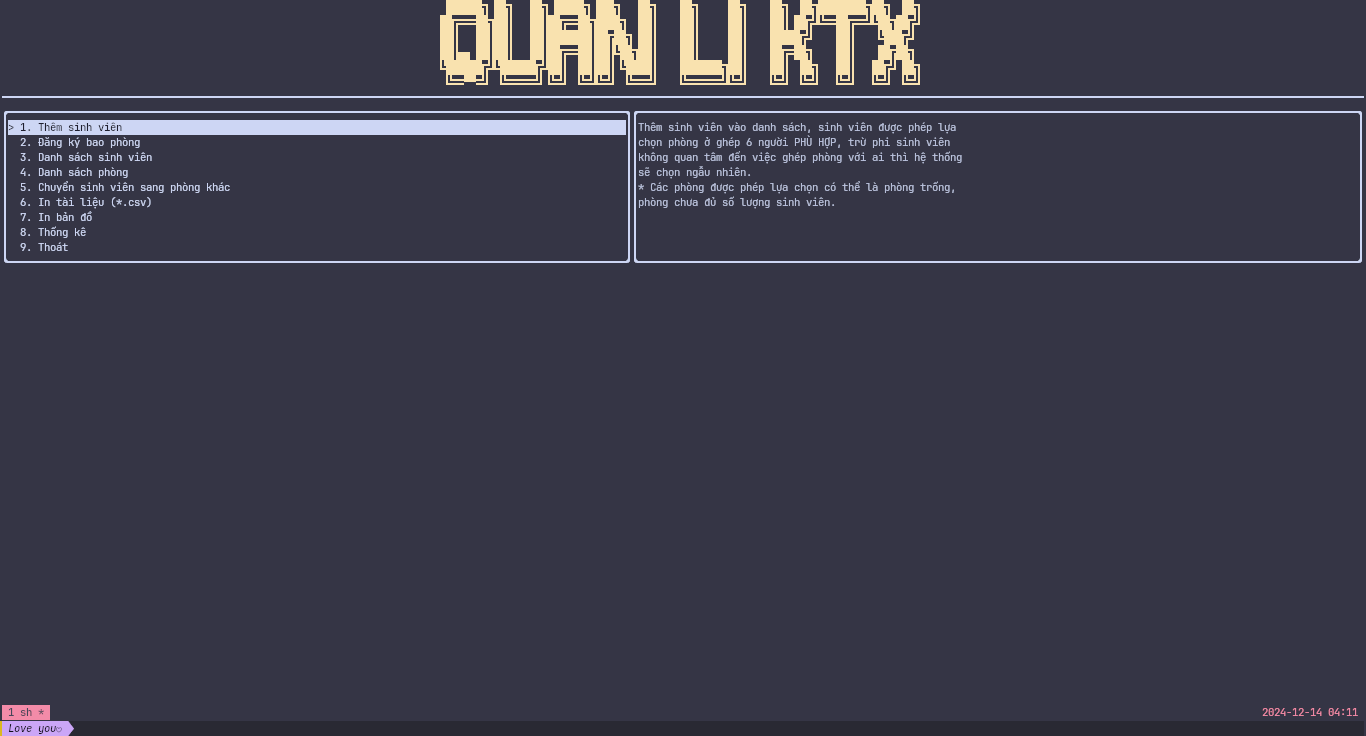
Màn hình welcome là giao diện đầu tiên mà người dùng truy cập vào phần mềm. chức năng này giúp tạo ấn tượng ban đầu cho người dùng nói chung và nhân viên quản lý ký túc xá nói riêng. Cụ thể, màn hình welcome bao gồm:

* Thông tin giới thiệu về nhóm nhà phát triển: tên nhóm, tên đồ án, khoa quản lý, tên ứng dụng, phiên bản ứng dụng.
* Thông tin giới thiệu về ký túc xá: Tên ký túc xá, địa chỉ ký túc xá.
* Thông tin giới thiệu về ứng dụng: ngôn ngữ lập trình và các thư viện, module được sử dụng trong quá trình phát triển phần mềm và đường dẫn git là nơi lưu trử source code của ứng dụng.
* Thông tin cụ thể nhà phát triển (About us): Danh sách tên nhà phát triển, mã số sinh viên, lớp, lớp học phần.

Ứng dụng hiện đang trong quá trình phát triển và hiện chỉ đang hỗ trợ quản lý sinh viên thuộc Ký túc xá phía tây thành phố Đà Nẵng. Hiện tại, bởi vì cấu trúc của các ký túc xá trên toàn thành phố Đà Nẵng nói riêng và trên cả nước nói chung tương đối khác nhau, nên ứng dụng chưa có giải pháp cụ thể để xây dựng cơ sở dữ liệu phù hợp.

Trong quá trình render màn hình màn hình welcome thì ứng dụng sẽ chạy đa luồng (multi-threading) với một tiểu trình khác là tải cơ sở dữ liệu và kiểm tra và tạo các khoản phí cho sinh viên nếu đến hạn, quá trình sẽ diễn ra trực quan và đẹp mắt dưới dạng thanh tiến trình như trên.

## Màn hình menu chính



*Hình 2.3: Màn hình menu chính*

Màn hình menu chính là giao diện trung tâm của hệ thống quản lý ký túc xá, nơi cung cấp phần lớn các chức năng chính lớn của phần mềm, giúp người dung dễ dàng truy cập vào các tính năng cần thiết của phần mềm. Màn hình menu chính hỗ trợ một số cơ chế đặc biệt thân thiện, dễ dàng sử dụng và hỗ trợ thao tác nhanh cho người dùng, như:

1. Cơ chế rê chuột để chọn tính năng: người dùng chỉ việc rê và kích chuột đến các tính năng cụ thể, sau đó gõ Enter để sử dụng tính năng.
2. Cơ chế sử dụng bàn phím số để chọn tính năng dựa trên chỉ số của tính năng hiển thị trên màn hình menu.
3. Cơ chế sử dụng phím mũi tên lên/xuống trên bàn phím để lựa chọn tính năng.

Ngoài ra, giao diện menu chính sẽ hiển thị các dòng giới thiệu và hướng dẫn của từng tính năng cụ thể, và hiển thị tất cả các tính năng nhỏ được tích hợp.

A screenshot of a computer

Description automatically generated

*Hình 2.4: Phần giới thiệu của tính năng Thêm sinh viên*

A screenshot of a computer

Description automatically generated

*Hình 2.5: Giới thiệu của tính năng Đăng ký bao phòng*

A screenshot of a computer

Description automatically generated

*Hình 2.6: Giới thiệu của tính năng Danh sách Sinh viên*

A screenshot of a computer

Description automatically generated

*Hình 2.7: Giới thiệu của tính năng Danh sách Phòng*

A screenshot of a computer

Description automatically generated

*Hình 2.8: Giới thiệu của tính năng In tài liệu*

A screenshot of a video game

Description automatically generated

*Hình 2.9: Giới thiệu của tính năng In bản đồ*

A screenshot of a computer

Description automatically generated

*Hình 2.10: Giới thiệu của tính năng Thống Kê*

## Tính năng thêm sinh viên

Chức năng thêm sinh viên cho phép người dùng (nhân viên quản lý ký túc xá) thực hiện công việc thêm thông tin của sinh viên mới vào hệ thống quản lý ký túc xá. Dữ liệu được lưu vào cơ sở dữ liệu và phục vụ cho các chức năng quản lý khác như tìm kiếm, chỉnh sửa, xóa và thống kê.

Chức năng thêm sinh viên sẽ thêm các thông tin như họ và tên, ngày sinh, giới tính, quê quán, trường, ngành học, số điện thoại, email như hình sau.

A screenshot of a video game

Description automatically generated

*Hình 2.11: Chức năng thêm sinh viên*

Nhân viên quản lý ký túc xá bắt buộc phải nhập đủ thông tin để xác thực sinh viên, nếu không, chương trình sẽ hiển thị yêu cầu như hình sau.

A screenshot of a computer

Description automatically generated

*Hình 2.12: Lỗi nếu người dùng nhập thiếu thông tin*

Nếu nhập thành công, chương trình sẽ tạo một mã sinh viên và chuyển sang giao diện chọn phòng cho sinh viên (áp dụng của tính năng **Chuyển sinh viên sang phòng khác, mục 2.4**).

## Tính năng chuyển sinh viên sang phòng khác

Chức năng chuyển sinh viên sang phòng khác cho phép người dùng (nhân viên quản lý ký túc xá) thực hiện thay đổi phòng cho một sinh viên (nếu sinh viên có nhu cầu muốn chuyển phòng) hoặc chọn phòng cho sinh viên mới.

Việc chuyển phòng được thực hiện bởi một cấu trúc dữ liệu SpinnerTree (giới thiệu ở **chương 3**), đảm bảo việc chọn phòng diễn ra phù hợp, không chọn những phòng đang bảo trì hoặc đã đủ số lượng sinh viên có trong phòng, cũng như là chọn phòng phù hợp với giới tính của sinh viên.

A screenshot of a computer

Description automatically generated

*Hình 2.13: Tính năng chọn phòng cho sinh viên*

Thông tin đầu vào được nhập như nhau:

1. Mã số sinh viên: mã số được cấp của sinh viên, được tự động nhập khi chuyển từ giao diện thông tin sinh viên (kết quả sau khi tìm kiếm sinh viên hoặc duyệt từ danh sách sinh viên) hoặc mã số được cấp cho sinh viên mới. Tại đây:
   1. Nếu điền đúng mã số sinh viên, hệ thống sẽ hiển thị thông tin sinh viên theo cơ chế thời gian thực (realtime) một cách tức thì, để người dùng nhìn thấy tổng quan về thông tin sinh viên hiện tại, phục vụ cho **tính phù hợp** khi chuyển phòng, cũng như là tránh sai sót trong quá trình xử lý.
   2. Nếu điền sai mã số sinh viên thì hệ thống sẽ không hiển thì thông tin sinh viên và nếu người dùng cố tình nhấn xác nhận thì sẽ nhận thông báo lỗi.
2. Chọn phòng: Bao gồm ba mục là chọn khu, chọn tầng, chọn phòng, được xây dựng trên cấu trúc dữ liệu SpinnerTree để hiển thị các phòng phù hợp với sinh viên, sau khi chọn phòng thành công, hệ thống sẽ hiển thị thông tin phòng và danh sách sinh viên đang ở trong phòng đó, nhằm chọn phòng **phù hợp** với sinh viên (vd: một số sinh viên không được ở chung phòng với sinh viên khác giới, hoặc một số sinh viên muốn được ở chung phòng với người cùng quê hoặc cùng trường,…).

A screenshot of a computer

Description automatically generated

*Hình 2.14: Chọn phòng cho sinh viên*

## Tính năng danh sách sinh viên

Chức năng danh sách sinh viên cho phép người dùng (nhân viên quản lý ký túc xá) xem, tìm kiếm (sử dụng realtime Search, giới thiệu ở **chương 3**), quản lý thông tin chi tiết và thanh toán tiền phòng cho tất cả sinh viên đang lưu trú tại ký túc xá. Danh sách sinh viên sẽ được sắp xếp sẵn theo thứ tự từ điển mỗi khi thêm sinh viên vào ký túc xá (sử dụng Insertion Sort, giới thiệu ở **chương 3**).

A screenshot of a computer

Description automatically generated

*Hình 2.15: Tính năng danh sách sinh viên*

A screenshot of a computer

Description automatically generated

*Hình 2.16: Cơ chế realtime search của danh sách sinh viên*

1. Ngoài ra còn có cơ chế hiển thị danh sách sinh viên đang nợ tiền phòng:

A screenshot of a computer

Description automatically generated

*Hình 2.17: Hiển thị danh sách sinh viên nợ tiền phòng*

1. Sau khi nhấn vào thông tin chi tiết thì hệ thống sẽ chuyển sang giao diện thông tin chi tiết của sinh viên như sau (bao gồm thông tin sinh viên, các tính năng xóa sinh viên, cập nhật thông tin sinh viên, đăng ký xe, thanh toán tiền phòng, đổi phòng và nút quay lại).

A screenshot of a computer

Description automatically generated

*Hình 2.18: Thông tin chi tiết của sinh viên*

1. Thanh toán tiền phòng:



*Hình 2.19: Thanh toán tiền phòng*

1. Đăng ký xe: hiện tại thì ứng dụng chỉ cho phép đăng ký tối đa một xe/một sinh viên, tiền phí xe được công vào tiền phòng của sinh viên, và được tính dựa theo loại xe mà sinh viên đăng ký (xe máy/xe đạp/ô tô).

A screenshot of a computer

Description automatically generated

*Hình 2.20: Đăng ký xe*

A screenshot of a computer

Description automatically generated

*Hình 2.21: Đăng ký xe thành công*

1. Cập nhật thông tin sinh viên: hệ thống sử dụng giao diện thêm sinh viên để thiết kế chức năng cập nhật thông tin sinh viên, thông tin sinh viên hiện tại được hiển thị sẵn trong các trường để cho người dùng chỉnh sửa.

A screenshot of a video game

Description automatically generated

*Hình 2.22: Cập nhật thông tin sinh viên*

Sau khi cập nhật xong thì hệ thống sẽ quay lại giao diện thông tin chi tiết sinh viên, lúc này thì thông tin sinh viên đã được cập nhật:

A screenshot of a video game

Description automatically generated

*Hình 2.23: Cập nhật thông tin sinh viên thành công*

1. Đổi phòng: thực hiện đổi cầu theo nhu cầu của sinh viên, hệ thống sẽ chuyển qua giao diện **chuyển sinh viên sang phòng khác (mục 2.4)**.
2. Xóa sinh viên: thưc hiện xóa sinh viên trong cơ sở dữ liệu khi sinh viên không còn ở trong ký túc xá nữa.

## Tính năng danh sách phòng

Chức năng danh sách phòng cho phép người dùng (nhân viên quản lý ký túc xá) xem, quản lý, theo dõi thông tin và thanh toán tiền điện cho từng phòng.

Giống với tính năng danh sách sinh viên thì tính năng danh sách phòng cũng hỗ trợ cơ chế realtime search.

A screenshot of a computer

Description automatically generated

*Hình 2.24: Cơ chế realtime search trong danh sách phòng*

1. Ngoài ra, còn có cơ chế hiển thị những phòng còn nợ tiền phòng.

A screenshot of a computer

Description automatically generated

*Hình 2.25: Hiển thị danh sách những phòng đang nợ tiền phòng*

1. Và cơ chế hiển thị những phòng đang bảo trì, để kịp thời nắm danh sách số lượng những phòng cần sửa chữa.

A screenshot of a computer

Description automatically generated

*Hình 2.26: Hiển thị danh sách những phòng đanag bảo trì*

1. Khi nhấn vào thêm chi tiết phòng thì ứng dụng sẽ hiển thi thông tin chi tiết của phòng, đồng thời hiển thi danh sách sinh viên đang lưu trú trong phòng hiện tại. Hỗ trợ các tính năng như **Đăng ký bao phòng (mục 2.7)**, thanh toán tiền điện, thông báo sửa chữa kịp thời và hiệu quả.

A screenshot of a computer

Description automatically generated

*Hình 2.27: Giao diện thông tin chi tiết phòng*

1. Chức năng đăng ký bao phòng: màn hình sẽ chuyển sang giao diện **Đăng ký bao phòng (mục 2.7).**
2. Chức năng thanh toán tiền điện: sau khi nhấn nút thanh toán thì hệ thống sẽ tiến hành quá trình thanh toán và màn hình sẽ hiển thị thông báo thanh toán thành công tương tự như thanh toán tiền phòng.
3. Chức năng thông báo sửa chữa: sau khi nhấn nút thông báo sửa chữa thì hệ thống sẽ di chuyển tất cả sinh viên có trong phòng ra ngoài, đồng thời thay đổi trạng thái phòng thành “Bảo trì” và nút thông báo sửa chữa sẽ chuyển thành “thông báo đã sửa chữa”, đảm bảo hệ thống hoạt động thân thiện và dễ sử dụng.

Hiện tại, ứng dụng vẫn chưa đi vào thực tiễn nên cơ chế tính tiền điện (thông thường sẽ thông qua một bộ vi điều khiển đọc giá trị của công tơ điện và gửi lên Webserver). Ứng dụng vẫn đang sử dụng một Webserver tạm thời (static web) để tính tiền điện phục vụ cho quá trình demo của chương trình (tại đường dẫn <https://antialberteinstein.github.io/ElectricityCalculatorForPBL2/>, source code: <https://github.com/antialberteinstein/ElectricityCalculatorForPBL2>).

## Tính năng đăng ký bao phòng

Tính năng đăng ký bao phòng cho phép người dùng (nhân viên quản lý ký túc xá) có thể hỗ trợ sinh viên thực hiện đăng ký một phòng là phòng bao, nghĩa là phòng có số lượng sinh viên nhỏ hơn sáu. Tính năng này thường áp dụng cho các nhóm sinh viên muốn cùng ở trong một phòng hoặc dành riêng phòng cho một mục đích cụ thể (như phòng chuyên dụng, phòng họp,..).

A screenshot of a computer

Description automatically generated

*Hình 2.28: Tính năng đăng ký bao phòng*

Tương tự như tính năng chuyển phòng thì chức năng này cũng sử dụng cơ chế real-time để cập nhật thông tin phòng đang được chọn nếu người dung thay đổi nội dung của trường ký tự.

Ở mục bên phải thì người dùng có thể nhập vào số lượng sinh viên ở trong phòng, ví dụ người dung muốn đăng ký bao phòng cho 3 sinh viên thì nhập ở đây là 3. Số lượng sinh viên phải nhỏ hơn sáu (số lượng tối đa sinh viên trong mỗi phòng) và phải lớn hơn số lượng sinh viên tại có trong phòng.

Sau khi đăng bao phỏng thành công thì màn hình sẽ chuyển sang giao diện trước đó, ở đây, hệ thống đang đăng ký bao phòng trực tiếp từ giao diện thông tin chi tiết của phòng nên màn hình chuyển về giao diện như hình dưới.

A screenshot of a computer

Description automatically generated

*Hình 2.29: Đăng ký bao phòng thành công*

## Tính năng in tài liệu

Tính năng in tài cho phép người dùng (nhân viên quản lý ký túc xá) in danh sách sinh viên và danh sách phòng dưới dạng file csv, người dùng có thể đồng thời kết hợp sử dụng phần mềm MS Excel, R Studio,… để tra cứu, phân tích dữ liệu sinh viên, dữ liệu phòng, vẽ biểu đồ, thực hiện một số chức năng bị hạn chế trên giao diện dòng lệnh để nhằm giúp việc quản lý sinh viên, phòng trở nên trực quan, dễ dàng và hiệu quả. Ngoài ra thì người dùng cũng có thể in trực tiếp danh sách trực tiếp ra khổ giấy A4 để phục vụ cho quá trình quản lý tốt hơn.

A screenshot of a video game

Description automatically generated

*Hình 2.30: Tính năng in danh sách sinh viên*

Khi người dung nhấn vào nút “Chọn file” thì màn hình sẽ hiển thị một hộp thoại dialog yêu cầu người dung chọn đường dẫn để lưu file csv.

A screenshot of a computer

Description automatically generated

*Hình 2.31: Chọn đường dẫn file*

Ở đây thì hộp thoại “Choose Directory” không thuộc phạm vi của đồ án, vì ứng dụng hiện chỉ hỗ trợ trên giao diện dòng lệnh nên ứng dụng tạm thời sử dụng file nhị phân được biên dịch sẵn để tạo hộp thoại “Choose Directory” (sử dụng thư viện Tkinter, đường dẫn tải về: <https://github.com/antialberteinstein/FileDialog/releases>, source code: <https://github.com/antialberteinstein/FileDialog>).

Sau khi chọn đường dẫn thành công và nhấn “OK”, thì đường dẫn lưu file sẽ hiển thị như hình bên dưới.

A screenshot of a computer

Description automatically generated

*Hình 2.32: Chọn file thành công*

Người dùng chỉ việc nhấn vào nút in ở bên dưới, hệ thống sẽ tiến hành tạo file DanhSachSinhVien.csv một cách tự động và hiện thông báo nếu như in thành công.

A screenshot of a video game

Description automatically generated

*Hình 2.33: In file thành công*

Danh sách sinh viên sau khi in:

A screenshot of a computer

Description automatically generated

*Hình 2.34: Danh sách sinh viên*

Tương tự với chức năng in danh sách phòng.

A screenshot of a video game

Description automatically generated

*Hình 2.35: In danh sách phòng*

Trong trường hợp các file trùng tên nhau thì hệ thống sẽ tự động thêm các hậu tố (1), (2),… vào đuôi file để dễ dàng phân biệt và tránh ghi đè file cũ, gây mất dữ liệu.

A screenshot of a computer

Description automatically generated

*Hình 2.36: Các file được thêm hậu tố để tránh ghi đè*

## Tính năng in bản đồ

Tính năng in bản đồ cho phép người dung (người quản lý ký túc xá) xuất ra bản đồ ký túc xá hiển thị bố trí các khu của ký túc xá, nhằm hỗ trợ tân sinh viên dễ dàng tìm kiếm khu mình lưu trú, đồng thời nắm bắt được các khu vực chức năng như căn tin, khu hành chính, …

Tính năng in bản đồ có giao diện khá tương đồng với tính năng in tài liệu.

A screenshot of a video game

Description automatically generated

*Hình 2.37: Tính năng in bản đồ*

A screenshot of a computer

Description automatically generated

*Hình 2.38: Kết quả của tính năng in bản đồ*

## Tính năng thống kê

Tình năng thống kê cho phép người dùng (nhân viên quản lý ký túc xá) có một cái nhìn trưc quan về những thống kê trong ký túc xá, bao gồm: tổng số lượng sinh viên, số lượng sinh viên theo giới tính, tổng số lượng phòng, số lượng phòng bảo trì, số lượng phòng có đủ sinh viên,… Tính năng này hỗ trợ trực quan hóa dữ liệu, giúp cho việc quản lý và đưa ra quyết định hiệu quả, kịp thời nắm bắt tình hình ký túc xá, đồng thời có thể áp dụng công nghệ AI để hỗ trợ đưa ra quyết định trong tương lai.

A screenshot of a video game

Description automatically generated

*Hình 2.39: Tính năng thống kê*

Ngoài ra thì trong chính tính năng thống kê, thì hệ thống còn hỗ trợ cơ chế tìm kiếm nhanh một sinh viên theo tên (bằng thuật toán **Binary Search, sẽ giới thiệu ở chương 3**), fast-search cho hiệu năng và tốc độ ấn tượng nhằm phục vụ cho nhu cầu khẩn cấp của sinh viên.

A screenshot of a video game

Description automatically generated

*Hình 2.40: Tính năng tìm kiếm nhanh*

Sau khi tìm kiếm thành công thì hệ thống sẽ hiển thị một thông báo ra màn hình và một nút nhấn để người dung chuyển sang giao diện thông tin chi tiết của sinh viên nhằm thực hiện các tính năng khác.

A screenshot of a computer

Description automatically generated

*Hình 2.41: Kết quả tìm kiếm*

# THIẾT KẾ CẤU TRÚC DỮ LIỆU

## Phát biểu bài toán

* **Input:**

Chương trình sử dụng hệ quản trị cơ sở dữ liệu leveldb (NonSQL), nên đầu vào của chương trình là hệ thống file của leveldb như hình:

A screenshot of a computer

Description automatically generated

*Hình 3.1: Cấu trúc hệ quản trị cơ sở dữ liệu leveldb*

A screenshot of a computer

Description automatically generated

*Hình 3.2: Cấu trúc file của một bảng trong leveldb*

Ngoài ra thì còn một số dữ liệu quan trọng khác như descriptions của menu, ascii arts tiêu đề, file bản đồ và hai file nhị phân folderdialog (hiển thị hộp thoại “Choose Directory”) và retrieve\_data (lấy dữ liệu tiền điện từ Webserver).

Và dữ liệu sinh viên được người dung trực tiếp nhập vào từ bàn phím.

* **Output:**

Đầu ra của chương trình là các tài liệu như danh sách sinh viên, danh sách phòng và bản đồ, cũng như là những dữ liệu hiển thị trên màn hình qua quá trình sử dụng ứng dụng như: danh sách sinh viên, danh sách phòng, thống kê,…

## Phân tích và ứng dụng cấu trúc dữ liệu trong hệ thống

* + 1. **Vector**

Hệ thống sử dụng cấu trúc dữ liệu Vector để thay thế cho cấu trúc dữ liệu mảng mặc định của C/C++ (vốn có hiệu quả sử dụng không cao).

Ưu diểm của cấu trúc dữ liệu Vector là truy cập nhanh, có kích thước động và bảo toàn thứ tự, rất thích để thay thế mảng.

Nhóm sử dụng template để dễ dàng áp dụng Vector cho mọi kiểu dữ liệu trong hệ thống và hầu hết các hàm theo tiêu chuẩn của std::vector.

|  |
| --- |
| template <typename T>  class Vector {  public:  Vector();  Vector(const Vector<T>& v);  Vector(T\* data);  Vector& operator=(const Vector<T>& v);  Vector& operator=(T\* data);  void push\_back(const T& value);  T pop\_back();  T at(size\_t index) const;  T& operator[](size\_t index);  size\_t size() const;  void clear();  private:  unique\_ptr<T[]> data\_;  size\_t size\_;  size\_t capacity\_;  void resize();  }; |

Ở đây thì unique\_ptr và shared\_ptr là những kiểu con trỏ thông minh (smart pointers), nhằm tự động thu hồi bộ nhớ sau khi sử dụng con trỏ, nhằm tối ưu hóa bộ nhớ.

* + 1. **List**

Tương tự với Vector thì nhóm cũng đồng thời sử dụng List để thay thế kiểu dữ liệu mảng.

Ưu diểm của List là dễ dàng xây dựng, truy cập nhanh và dễ dàng duyệt các phần tử vì dữ liệu được lưu trữ liên tiếp trong bộ nhớ, thích hợp để lưu trữ danh sách sinh viên, danh sách phòng.

Nhóm sử dụng template để dễ dàng áp dụng List cho tất cả các kiểu dữ liệu trong hệ thống (đặc biệt là Student và Room).

|  |
| --- |
| template <typename T>  class List {  public:  List(int capacity=DEFAULT\_LIST\_CAPACITY);  List(const List& l);    List& operator=(const List& l);  List& operator=(const T\* arr);  void add(T item);  void remove(int index);  void clear();  size\_t size();  T& operator[](int index);  private:  unique\_ptr<T[]> list\_;  size\_t size\_;  size\_t capacity\_;  }; |

* + 1. **Date**

Cấu trúc dữ liệu Date là cấu trúc dữ liệu quan trọng được hệ thống sử dụng trong quá trình lưu trữ và tính toán tiền điện, tiền phòng cho sinh viên.

|  |
| --- |
| // Date class support Gregorian date format.  class Date {  public:  Date(int day, int month, int year);  int get\_day() const;  int get\_month() const;  int get\_year() const;  Date& operator++();  Date operator++(int);  Date& operator+=(int days);  Date operator+(int days) const;  bool operator==(const Date& other) const;  bool operator!=(const Date& other) const;  bool operator<(const Date& other) const;  bool operator<=(const Date& other) const;  bool operator>(const Date& other) const;  bool operator>=(const Date& other) const;  // Gregorian date format: DD/MM/YYYY  std::string to\_string() const;  static Date today();  private:  int day;  int month;  int year;  }; |

* + 1. **DateConverter**

DateConverter là một bộ chuyển đổi giúp hệ thống chuyển đổi giữa định dạng ngày tháng dd/MM/YYYY (định dạng Grenadian, sử dụng phổ biến tại Việt Nam) - định dạng YYYY-MM-dd (định dạng chuẩn quốc tế ISO) – chuỗi ký tự, nhằm hỗ trợ một số hệ thống lưu trữ không cho phép định dạng dd/MM/YYYY mà vẫn thân thiện với phần đông người dùng tại Việt Nam.

Các phương thức được định nghĩa toàn cục trong DateConverter:

1. ‘from\_string()’: chuyển từ chuỗi định dạng YYYY-MM-dd sang Date, độ phức tạp O(1).
2. ‘to\_string()’: chuyển từ Date sang chuỗi định dạng YYYY-MM-dd, độ phức tạp O(1).
3. ‘from\_grenadian\_string()’: chuyễn từ chuỗi định dạng dd/MM/YYYY sang Date, độ phức tạp O(1).
4. ‘to\_grenadian\_string()’: chuyển từ Date sang chuỗi định dạng dd/MM/YYYY, độ phức tạp O(1).
   * 1. **StringAdapter**

StringAdapter là bộ mang một Vector của kiểu dữ liệu chuỗi mà hệ thống dung để tăng tính tương thích của Vector với một số hàm của thư viện đồ họa chỉ cho phép sử dụng std::vector, tái sử dụng mã nguồn, tang tính linh hoạt và giảm sự phụ thuộc.

|  |
| --- |
| class StringAdapter : public ftxui::ConstStringListRef::Adapter {  public:  static std::unique\_ptr<StringAdapter> From(Vector<std::string>\* entries) {  return std::make\_unique<StringAdapter>(entries);  }  explicit StringAdapter(Vector<std::string>\* entries) : entries\_(entries) {}  ~StringAdapter() final = default;  size\_t size() const final { return entries\_->size(); }  std::string operator[](size\_t index) const final { return entries\_->at(index); }  private:  Vector<std::string>\* entries\_;  }; |

* + 1. **Queue**

Hệ thống sử dụng Queue để lưu trữ các mã số của sinh viên đã bị xóa nhằm tái sử dụng mã số cho sinh viên khác trong tương lai, tránh trường hợp không đủ số lượng mã số được cấp cho sinh viên.

Nhóm sử dụng template để dễ dàng áp dụng Queue cho mọi kiểu dữ liệu trong hệ thống.

|  |
| --- |
| template <typename T>  class Queue {  public:  Queue(int capacity = DEFAULT\_QUEUE\_CAPACITY);  Queue(const Queue& l);  void enqueue(T item);  T dequeue();  bool is\_empty() const;  bool is\_full() const;  private:  unique\_ptr<T[]> data\_;  size\_t front\_;  size\_t rear\_;  size\_t capacity\_;  void increment(size\_t& s);  }; |

* + 1. **Set**

Trong hệ thống của nhóm, Set là lớp được xây dựng với ba phương thức toàn cục union (hợp), intercept (giao) và except (trừ) thực hiện trên cấu trúc dữ liệu Vector nhằm thực hiện một số cơ chế của chương trình.

1. ‘union()’: phép hợp, độ phức tạp .
2. ‘intercept()’: phép giao, độ phức tạp .
3. ‘except()’: phép trừ, độ phức tạp .
   * 1. **SpinnerTree**

Trong quá trình xây dựng và thiết kế hệ thống, nhóm đã đề ra một cấu trúc dữ liệu mới là SpinnerTree nhằm phục vụ cho quá trình chọn phòng diễn ra hiệu quả và phù hợp.

Cụ thể, SpinnerTree là một cấu trúc cây có root là một chuỗi là tên của cây, các node cây được phân chia thành ba tầng:

* Tầng 1 (sau root): là các node lưu trữ tên của các khu còn phòng trống, tất cả là con của root.
* Tầng 2 (sau tầng 1): là các node lưu trữ tên của các tầng còn phòng trống tương ứng với mỗi khu và là con của các node tầng 1.
* Tầng 3 (sau tầng 2): là các node lưu trữ tên của các phòng trống tương ứng với mỗi tầng và là con của các node tầng 2.

Hệ thống cung cấp một hàm là spin() trả về root của SpinnerTree để phục vụ cho quá trình chọn phòng.

*ADD\_CHILD(node, value):*

*IF value là con của node :*

*return value*

*node.child .append(value)*

*return node.child.last*

*SPIN():*

*Cho A là danh sách phòng.*

*n:= A.length*

*Cho root lưu trữ một chuỗi , không có node con.*

*FOR i := 0 to n:*

*IF A[i] .status = “Còn trống”:*

*child := ADD\_CHILD(root, A[i].block)*

*child := ADD\_CHILD(child, A[i].floor)*

*child := ADD\_CHILD(child, A[i].room)*

*return root*

Hàm ADD\_CHILD có độ phức tạp thuật toán O(1).

Hàm SPIN có độ phức tạp thuật toán O(n), độ phức tạp không gian là O(n) (có tối đa n nodes ở tầng 3).

* + 1. **Scroller**

Scroller là cấu trúc dữ liệu được xây dung để hỗ trợ hiển thị danh sách sinh viên và danh sách phòng dưới dạng bảng, hỗ trợ hiển thị lượng lớn dữ liệu trong một không gian giới hạn, do thư viện giao diện mà nhóm đang sử dụng chưa hỗ trợ cơ chế scroller. Người dùng có thể dùng các phím điều hướng để xem toàn bộ danh sách mà không cần làm cho giao diện quá phức tạp hoặc rối mắt.

Các tác vụ sử dụng của Scroller có độ phức tạp O(k), trong đó k là số phần tử được hiển thị ra màn hình, thực tế thì giá trị của k nhỏ hơn rất nhiều so với n (số lượng phần tử tối đa), cho nên độ phức tạp của các tác vụ này thường được xem là tương đương với O(1).

Các tác vụ tạo Scroller giống như thêm các trường (trong dự án nhóm gọi nó là một ScrollerMap, map giữa hai kiểu dữ liệu là String và List), tức là các cột dữ liệu trong bảng, tốn O(n). Tác vụ thêm các record mới vào Scroller tốn O(t x n), trong đó t là số trường trong Scroller, số trường của Scroller cũng rất ít khi so với n, nên nhóm xem độ phức tạp của tác vụ này cũng là O(n), khi mà tác vụ push\_back() của Vector có độ phức tạp O(1) đối với mỗi lần thêm.

A diagram of a computer program

Description automatically generated

*Hình 3.3: Cấu trúc dữ liệu Scroller*

Trong đó thì CompleteList là một List lưu trữ danh sách hoàn thiện, có độ phức tạp không gian O(n). DisplayList là List lưu trữ danh sách các phần tử được hiển thị trên màn hình, có độ phức tạp không gian O(k). BackupList là List lưu trữ thông tin tương tự như CompleteList, trong cơ chế realtime search, CompleteList sẽ bị thay đổi để phù hợp với danh sách đã được lọc, trong khi BackupList vẫn giữ nguyên danh sách hoàn thiện và được gán khôi phục cho CompleteList sau khi kết thúc realtime search. Scroller bao gồm các ScrollerMap được hiển thị ra màn hình dưới dạng các cột và sử dụng cơ chế đồng bộ để tránh các dữ liệu hiển thị ở sai dòng.

* + 1. **CSVCreator**

CSVCreator là trình tạo file csv, nhằm xuất ra các dữ liệu như danh sách sinh viên, danh sách phòng dưới dạng csv.

Tương tự với Scroller thì CSVCreator cũng bao gồm các CSVCreatorMap (map giữa tên cột và một List dữ liệu).

Các tác vụ khởi tạo như thêm các trường tốn O(t), thêm các record tốn O(t x n). Các tác vụ sử dụng như ghi dữ liệu vào file tốn O(t x n), nếu như ta xem như tác vụ ghi dữ liệu vào mỗi ô là O(1). Tương tự thì tác vụ đọc dữ liệu từ file có độ phức tạp thuật toán O(t x n).

Khác với ScrollerMap sử dụng ba List, thì CSVCreatorMap chỉ sử dụng duy nhất một List để lưu dữ liệu, do đó nó có độ phức tạp không gian cho mỗi CSVCreatorMap là O(n).

* + 1. **ModelProducer**

ModelProducer, có thể được xem như phần backend của dự án, nó nhằm quản lý các model và giao tiếp giữa giao diện và cơ sở dữ liệu, mỗi ModelProducer gắn với mỗi thực thể đươc lưu trong cơ sở dữ liệu. Nhược điểm của ModelProducer là kiến trúc phức tạp, khó bảo trì do nhóm có ít thời gian để nghiên cứu và tối ưu, tuy nhiên, nhóm đã sử dụng Factory Design Pattern để khắc phục nhược điểm này của ModelProducer.

ModelProducer đươc xây dựng dựa trên các phương thức chính:

+) get\_all\_keys(): lấy toàn bộ key của dữ liệu trong cơ sở dữ liệu, key được hashing hai lần, một lần từ client và một lần từ leveldb. Phương thức này dựa trên cờ  *student\_sorted\_flag,* nếu cờ bật, get\_all\_keys() sẽ tiến hành sort danh sách keys bằng InsertionSort theo tên của sinh viên, dĩ nhiên cờ này chỉ hoạt động trên ModelProducer của thực thể sinh viên. Trong trường hợp cờ tắt, độ phức tạp của nó là O(n), trường hợp còn lại thì độ phức tạp của nó cộng thêm với độ phức tạp của InsertionSort.

+) add(): Thêm một thực thể vào cơ sở dữ liệu, nhưng ta được biết, leveldb lưu trữ dữ liệu thông qua cặp key-value, cho nên quá trình này trong thực tế là quá trình gán thực thể với một key, được sinh ra bởi HashFunction (**Mục 3.2.16**). Độ phức tạp của nó là O(1).

+) remove(): Xóa một thực thể khỏi cơ sở dữ liệu, tương tự thì quá trình này chỉ xóa key của nó khỏi leveldb. Độ phức tạp của nó là O(1).

+) get(): Truy cập một thực thể từ cơ sở dữ liệu từ key có sẵn, có độ phức tạp là O(1).

+) modify(): Chỉnh sửa một thực thể có sẵn trong cơ sở dữ liệu qua key có sẵn. Quá trình này tương tự với add(), nhưng nếu key tồn tại thì nó sẽ ghi đè giá trị vào key hiện có, thực thể chỉnh sửa là một thực thể mới hoàn toàn và tất cả thông tin của nó sẽ được thay thế với thực thể cũ tại key đó. Độ phức tạp của modify() là O(1).

Ngoài ra thì ModelProducer còn có một số phương thức khác phục vụ cho việc xây dựng và quản lý những lớp thực thể, và những phương thức này có độ phức tạp là O(1). Tương tự như những thư viện hỗ trợ cơ sở dữ liệu khác thì ModelProducer sử dụng Lazy Singleton để tối ưu hóa công việc.

Cấu trúc ModelProducer là cấu trúc dữ liệu quan trọng nhất trong chương trình của nhóm, mặc dù được xây dựng phức tạp nhưng nhờ sử dụng các Design Pattern như Lazy Singleton hay Factory và các cờ hiệu thì nó đạt được hiệu năng vô cùng tối ưu, gần tương đương với các thư viện hàng đầu về cơ sở dữ liệu như ORM,… (mặc dù ModelProducer ít phức tạp hơn).

* + 1. **FeeCalculator**

FeeCalculator là phần backend thứ hai của dự án, nhiệm vụ của nó là tạo và quàn lý các khoản chi phí trong ký túc xá, ở đây là tiền phòng và tiền điện. Tương tự như ModelProducer thì FeeCalculator cũng được xây dựng từ Lazy Singleton và Factory.

Đối với đối tượng FeeCalculator xây dựng cho tiền phòng của sinh viên, trong đó tiền phòng thường được tính theo công thức:

Trong đó:

P là tiền phòng của sinh viên.

là khoảng chi phí tối đa mà sinh viên phải trả.

N là số sinh viên tối đa ở trong phòng.

n là số sinh viên hiện tại đang lưu trú trong phòng.

V là tiền phí gửi xe của sinh viên.

m là số tháng đóng.

Tác vụ tạo khoản thanh toán có cấu trúc sơ lược là: nếu đến ngày nộp phí (thời điểm cuối cùng mà sinh viên đóng phí cộng với số tháng đóng m), hệ thống sẽ gọi một HashFunction (client-level) để ghi đè dữ liệu tiền phòng trước đó (nếu đã đóng tiền), dĩ nhiên là dữ liệu trước đó không bị mất và có thể được gọi trở lại phục vụ việc tra cứu bằng HashFunction. Key của thực thể tiền phòng cũng được Hash theo mã sinh viên tương ứng, mặc dù thưc thể tiền phòng có thuộc tính mã sinh viên, nhưng trong việc truy cập từ thực thể sinh viên, quá trình này sẽ tiêu tốn O(n), cơ chế Hash sẽ làm giảm độ phức tạp về O(1). Nếu quá một khoảng thời gian nào đó mà sinh viên đó chưa đóng tiền phòng, chương trình sẽ chuyển trạng thái của khoản thanh toán thành “Quá hạn”, và sinh viên đó bị buộc rời khỏi phòng, hệ thống sẽ không tính tiền phòng của sinh viên đó trong tương lai.

Đối với đối tương FeeCalculator phục vụ xây dựng tiền điện cho từng phòng, trong đó tiền điện được tính theo công thức *số chữ điện x giá mỗi chữ điện (VND)*, hệ thống sử một cơ chế giống như API (RetrieveData) để lấy dữ liệu về số chữ điện phòng đã dùng trong tháng (đã giới thiệu ở phần trước) và tính toán tiền điện, cách thức tạo hóa đơn và Hash mã hóa đơn tương tự như đối với tính tiền phòng, nhưng khoản thanh toán tiền điện sẽ được Hash theo mã phòng.

Cấu trúc FeeCalculator là cấu trúc dữ liệu tương đối cần thiết trong dự án, nó đươc tối ưu nhờ cơ chế hashing của model (**giới thiệu ở phần 3.2.16**) và sử dụng ModelProducer làm nền tảng quản lý hai thực thể tiền phòng và tiền điện.

* + 1. **RegexingPattern**

RegexingPattern được sử dụng trong ModelProducer của sinh viên phục vụ cho việc sắp xếp, công dụng của nó là chia tên sinh viên thành hai phần họ và tên, nó sử dụng phương thức regex\_name() để chuyển từ một chuỗi thành một RegexingPattern theo quy tắc đặt tên của người Việt Nam.

Hàm regex\_name() chia tên đầy đủ của một người thành hai phần, last\_name và first\_name. Trong đó first\_name là từ nằm cuối trong tên và last\_name là những phần còn lại của tên. Hàm có độ phức tạp thuật toán là O(n), trong đó n là số ký tự trong tên đầy đủ của sinh viên.

RegexingPattern đươc xây dựng một phương thức so sánh phục vụ cho InsertionSort. Phương thức sẽ so sánh first\_name trước rồi mới đến last\_name.

* + 1. **SearchingRegex (thuật toán)**

SearchingRegex được sử dụng trong cơ chế realtime search. Nó loại bỏ tất cả các khoảng trống trong chuỗi và đưa tất cả các ký tự về dạng in thường phục vụ cho việc tìm kiếm dễ dàng hơn.

Độ phức tạp thuật toán là O(n), trong đó n là số ký tự của chuỗi tìm kiếm.

* + 1. **InsertionSort (thuật toán)**

InsertionSort sử dụng trong ModelProducer để sắp xếp danh sách sinh viên theo RegexingPattern được tạo bởi tên của sinh viên.

Thuật toán có mã giã như sau:

*INSERTION-SORT(A):*

*n := A.length*

*Let regexes be an array 1..n storing RegexingPattern of all students’ name*

*FOR i := 2 to n:*

*j := i – 1*

*while j >= 0 and regexes[i] < regexes[j]:*

*exchange A[i+1] and A[j]*

*exchange regexes[i+1] and regexes[j]*

*A[j + 1] = A[i]*

*regexes[j + 1] = regexes[i]*

Trên lý thuyết, InsertionSort có độ phức tạp , nhưng trong chương trình thì danh sách sinh viên vốn đã được sắp xếp sẵn mỗi khi thêm một sinh viên mới vào nên dô phức tạp trên thực tế của nó chỉ dao động từ O(1) đến O(n), nếu không xét đến độ phức tạp của regex. Đây cũng là lý do mà nhóm sử dụng InsertionSort để sắp xếp danh sách sinh viên.

* + 1. **HashFunction (hàm)**

Cơ chế Hash và các hàm Hash được sử dụng để tối ưu hóa hoạt động của chương trình. Mặc dù nhóm vẫn chưa sử dụng đến HashTable, hay những hàm Hash tiêu chuẩn, nhưng những hàm Hash được xây dựng trong chương trình vẫn đảm bảo tính chất của hashing, đặc biệt là chúng phù hợp với hoàn cảnh sử dụng trong phần mềm. Một số hàm Hash nhóm đã sử dụng:

1. Hash mã sinh viên: mã sinh viên là một chuỗi sáu chữ số được tạo từ số lượng sinh viên hiện tại của ký túc xá cộng thêm 1, trong một số trường hợp, mã sinh viên sẽ được lấy từ hàng đợi, đồng thời chương trình sử dụng thêm cơ chế Hash mã sinh viên từ tên sinh viên (*mã sinh viên := (mã sinh viên x 31 + mã từng ký tự) mod 100000007*).
2. Hash mã phòng: mã phòng là một chuỗi gồm bốn ký tự được Hash theo công thức  *mã phòng := mã khu + mã tầng \* 100 + mã phòng.*
3. Hash mã tiền phòng: mã tiền phòng sẽ được Hash bằng vói mã sinh viên của sinh viên tương ứng đối với khoản thanh toán mới nhất, các hóa đơn cũ sẽ được được Hash *mã tiền phòng := mã sinh viên + ‘\_’ + mã tiền phòng****.***
4. Hash mã tiền điện: tương tự với mã tiền phòng và được Hash dựa trên mã phòng.
5. Hash mã xe: mã xe được Hash bằng với mã sinh viên.
   * 1. **Cơ chế realtime search**

Cơ chế realtime search được sử dụng trong Scroller để hiển thị tên của sinh viên hoặc tên phòng từ chuỗi tìm kiếm một cách trực quan và ngay lập tức trưng bày ra màn hình những sinh viên hoặc phòng có tên tương ứng. Tập hợp tên sinh viên hoặc tên phòng (ký hiệu S, với các phần tử ) và chuỗi tìm kiếm (ký hiệu p) sẽ được regex bằng SearchingRegex thành S’ và p’. Hệ thống sẽ kiểm tra nếu thì , với R là tập kết quả sẽ được hiển thị trên màn hình.

Do tính chất đăc thù của realtime search nên nhóm chỉ có thể sử dụng phương pháp Linear để duyệt qua từng phần tử của danh sách với độ phức tạp O(n), kém tối ưu hơn thuật toán BinarySearch. Mặc dù vậy, kết quả thực thi và hiệu năng của cơ chế realtime search vẫn vô cùng ấn tượng và đạt được mục đích ban đầu của nhóm.

* + 1. **Thuật toán BinarySearch**

Thuật toán BinarySearch được sử dụng trong mục **Thống kê**, là cơ chế tìm kiếm nhanh được tích hợp trong mục này nhằm phục vụ trường hợp khẩn cấp.

Về mặt lý thuyết thì BinarySearch tối ưu hơn LinearSearch sử dụng trong cơ chế Realtime Search với độ phức tạp O(logn), nhất là đối với bộ dữ liệu đã được sắp xếp sẵn như danh sách sinh viên. Mã giả của thuật toán:

*BINARY\_SEARCH(name, regexes):*

*// Với regexes là tập RegexingPattern của danh sách sinh viên.*

*n := regexes.length*

*low := 1*

*high := n*

*while low <= high:*

*mid := low + (high – low)/2*

*if (regexes[mid]=name) then return mid*

*if (regexes[mid] < name) then low := mid + 1*

*else high := mid – 1*

*return NOT\_FOUND // -1*

# PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG VÀ TRIỂN KHAI HỆ THỐNG

## Cấu trúc hệ thống hướng đối tượng

* + 1. Cấu trúc hệ thống hướng đối tượng ở phần xử lý

A diagram of a computer program

Description automatically generated with medium confidence

*Hình 4.1: Cấu trúc hướng đối tượng ở phần xử lý*

Phần xử lý có thể được xem như phần cấu trúc xương sống của phần backend cũng như là bản thiết kế cơ sở dữ liệu hoàn hảo cho dự án. Bản thiết kế này bao gồm tám lớp đối tượng, một lớp trừu tượng và năm lớp enum. Trong đó, hệ thống xây dựng các bảng cơ sở dữ liệu cho năm lớp dẫn xuất cuối, tức là lớp không còn lớp con nào và là lớp xây dựng cụ thể nhất của chương trình, bao gồm các lớp Student, Room, ElectricityFeePayment, RoomFeePayment, Vehicle.

Lớp cha tối cao nhất, tức là lóp không còn lớp nào khác là cha của nó, là một lớp trừu tượng Trigger, nó cung cấp các bản dựng của các listener và liên kết với ModelProducer (liên kết từ lớp con Model) hỗ trợ chương trình hoạt động tương tự như cơ chế Trigger trong hệ quản trị cơ sở dữ liệu SQL, các listener của nó được triển khai như sau:

+) onPrepareToAdd(): được thực thể gọi trước khi thêm bản thân vào cơ sở dữ liệu thông qua ModelProducer, nó tạo một khoảng thời gian để cho thực thể xử lý các tác vụ cần thiết trước khi đi vào cơ sở dữ liệu.

+) on\_add(): được thực thể gọi sau khi thêm bản thân vào cơ sở dữ liệu thông qua ModelProducer, nhằm thực hiện các công việc dọn dẹp hoặc liện kết cập nhật dữ liệu ở các bảng khác.

+) on\_modify(): được thực thể gọi sau khi sửa đổi thông tin bản thân cơ sở dữ liệu thông qua ModelProducer, công dụng của nó tương tự với on\_add().

+) on\_remove(): được thực thể gọi sau khi xóa bản thân khỏi cơ sở dữ liệu thông qua ModelProducer, công dụng của nó tương tự on\_add() và on\_modify().

Kế sau Trigger là lớp Model, nó là lớp liên kết trực tiếp với ModelProducer, đóng vai trò là sản phẩm tổng quát của Factory giúp sản xuất các lớp con thông qua upcasting và dynamic binding. Trong đó ModelProducer sẽ đồng sử dụng lớp enum ModelType với ModelNameFactory, một nhà máy sản xuất tên của các lớp dưới dạng chuỗi phục vụ cho quá trình lưu trữ cơ sở dữ liệu, để nhận dạng sản phẩm cần tạo. Model dựng một số phương thức sau:

+) serialize(): phương thức chuyển thực thể thành một chuỗi json, phục vụ cho việc lưu trữ thực thể vào cơ sở dữ liệu leveldb dưới dạng value.

+) deserialize(): phương thức chuyển một chuỗi json thành thực thể tương ứng, phục vụ cho việc tải thực thể từ cơ sở dữ liệu.

+) hash\_to\_id(): hàm Hash chính của từng thực thể.

+) generate\_id(): hàm sinh mã của thực thể từ hàm Hash, phục vụ cho quá trình tạo và thêm Model vào cơ sở dữ liệu dưới dạng key.

Student là lớp kế thừa của Model, là lớp quan trọng nhất trong phần xử lý của dự án, nó cung cấp các thuộc tính cơ bản của một sinh viên như hình vẽ. Để tối ưu hóa hoạt động thêm sinh viên, nhóm tạo một cờ creating cho Student, cờ này giương lên khi client yêu cầu tạo một mã sinh viên mới, trong khi đó, Student sẽ sử dụng một mã sinh viên có sẵn nếu cờ này tắt. Thực tế, trong hầu hết trường hợp thêm sinh viên, mã sẵn có của lớp Student không tồn tại, do đó nó sẽ được tạo thông qua hàm Hash *mã sinh viên := (mã sinh viên x 31 + mã từng ký tự) mod 100000007* trong trường hợp client quên bật cờ.

Lớp quan trọng thứ hai trong phần này là Room, ngoài những thuộc tính cơ bản của một phòng thì Room sử dụng một một lớp enum là RoomStatus để lưu trữ trạng thái của phòng.

Gần giống Student thì Payment cũng sử dụng một cờ overridden, cờ này cũng được bật lên từ client để thông báo tạo một khoản thu mới, khi đó hàm Hash của Payment sẽ tạo mã thanh toán từ mã sinh viên, trong trường hợp ngược lại, hàm Hash sẽ tạo mã thanh toán từ mã sinh viên và ngày tạo. Cờ overridden được sử dụng khi client muốn lưu trữ khoản thu cũ (tắt overridden trước khi gọi add()) và tạo một khoản thu mới (bật overridden).

Payment cũng đồng thời một enum class là PaymentStatus lưu trữ trạng thái của hóa đơn.

Kế thừa Payment là hai lớp ElectricityFeePayment (dùng để tính tiền điện) và RoomFeePayment (dùng để tính tiền phòng), trong đó, RoomFeePayment sử dụng thêm dữ kiện từ VehicleFee để tính theo công thức đã nêu ở trên.

Lớp Vehicle lưu trữ các phương tiện mà sinh viên sử dụng và cất trữ trong ký túc xá, nó cũng sử dụng cờ creating nhưng có công dụng gần giống với cờ overridden để hỗ trợ lưu trữ nhiều hơn một xe cho sinh viên. Tuy nhiên tính năng này hiện đang bảo trì và chưa thể thực hiện được trong dự án của nhóm, nhưng mặt bằng chung thì chương trình đã tối ưu cho việc thêm mỗi sinh viên một xe và trong thực tế, hầu hết sinh viên thường chỉ đăng ký một xe. Lớp Vehicle sử dụng enum class VehicleType để lưu trữ loại xe.

Các mối liên kết (references) được biểu diễn dưới dạng dấu mũi tên có đỉnh được tô kín như hình vẽ. Các mối liên kết này thường được xử lý bằng Trigger và HashFunction trong dự án.

* + 1. Cấu trúc hướng đối tượng ở phần hiển thị

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

## Hướng phát triển

TÀI LIỆU THAM KHẢO

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

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