

Trường Đại học Khoa học Tự nhiên – ĐHQG TPHCM

Khoa Công nghệ Thông tin



PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

BÁO CÁO ĐỒ ÁN

CHỦ ĐỀ PHẦN MỀM QUẢN LÝ

ỨNG DỤNG BILY FLOWER SHOP

Giảng viên hướng dẫn: ThS. Trần Anh Duy

Nhóm sinh viên thực hiện:

Hoàng Trọng Vũ 20120025

Nguyễn Thiên An 20120030

Dũ Quốc Huy 20120101

Lớp: 20CTT1TN1

Thành phố Hồ Chí Minh, tháng 1 năm 2022

Lời nói đầu

Lập trình hướng đối tượng là một phương pháp lập trình có ứng dụng rộng rãi trong xây dựng và phát triển các sản phẩm phần mềm. Ra đời từ những năm 1960, lập trình hướng đối tượng ngày càng phát triển mạnh mẽ và cho ra đời các sản phẩm phần mềm chất lượng, mã nguồn dễ bảo trì và có khả năng tái sử dụng cao.

Từ những ưu điểm trên, chúng em muốn áp dụng những kiến thức và kỹ năng đã được học về lập trình hướng đối tượng để giải quyết các bài toán thực tế. Sau khoảng thời gian tìm tòi, nghiên cứu, chúng em đến với ý tưởng xây dựng một ứng dụng quản lý cho cửa hàng hoa **Bily** – một thương hiệu giả định đang có nhu cầu mở rộng quy mô kinh doanh lên sàn thương mại điện tử.

Trong quá trình phát triển ứng dụng, chúng em đã cố gắng thêm thắt, cải tiến một số chức năng và tiện ích để tận dụng tối đa các tính chất ưu việt của lập trình hướng đối tượng, bao gồm tính đóng gói, kế thừa và đa hình. Chúng em cũng cân nhắc trong việc thiết kế, đặt tên các lớp đối tượng sao cho trực quan, dễ hiểu nhất. Sơ đồ UML được thiết kế rõ ràng, nhất quán với mã nguồn chương trình và được phân chia rõ thành 3 miền với 3 màu khác nhau, ứng với các *lớp giao diện (GUI)*, *Providers* và các *lớp thành phần (Components)*, hạn chế tối đa việc giao nhau giữa các mối quan hệ đồng mức. Trong đó, các lớp *Provider* được cài đặt theo mẫu thiết kế **Singleton**, với vai trò là đồng thời “giả lập” các cơ sở dữ liệu cho ứng dụng và cung cấp một số thao tác liên quan giữa các lớp thành phần.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Minh Huy và thầy Trần Anh Duy, hai giảng viên của bộ môn và cũng là người tạo cho chúng em niềm đam mê với môn học này. Sự giảng dạy tận tình, tâm huyết cùng những hướng dẫn quý báu của thầy đã giúp chúng em có đủ kiến thức và kỹ năng, từ đó biết cách vận dụng vào hoàn thành tốt đồ án được giao.

Mặc dù đã có nhiều cố gắng, nỗ lực khi xây dựng ứng dụng, song chắc chắn không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp và phê bình từ phía Thầy để đồ án được hoàn thiện hơn.

Lời cuối cùng, chúng em xin kính chúc thầy nhiều sức khỏe, thành công và hạnh phúc.

Nhóm 6

MỤC LỤC

Lời nói đầu	2
I. Giới thiệu	6
II. Các chức năng cơ bản	7
II.1. Xem hoa theo danh mục, tìm kiếm hoa	7
II.2. Đăng ký, đăng nhập.....	7
II.3. Hệ thống tài khoản	8
II.3.1. Trang quản lý tài khoản	9
II.3.2. Ví điện tử Bily	9
II.3.3. Người mua	10
II.3.3.1. Sử dụng giỏ hàng	10
II.3.3.2. Đặt hàng, hủy đơn hàng	10
II.3.4. Người bán	12
II.3.4.1. Quản lý gian hàng	12
II.3.4.2. Quản lý các đơn hàng	13
II.3.4.3. Quản lý voucher	13
II.3.4.4. Thống kê dữ liệu	14
II.3.5. Nhân viên	15
II.3.5.1. Tiền lương và tiền thưởng.....	15
II.3.5.2. Quản lý sự kiện	15
II.3.6. Quản lý cửa cửa hàng.....	16
II.3.6.1. Quản lý các nhân viên.....	16
II.3.6.2. Quản lý hợp đồng người bán	17
II.3.6.3. Quản lý các sự kiện trong cửa hàng	18
III. Thiết kế đồ án	19
III.1. Tổ chức dữ liệu	19
III.2. Các lớp thành phần.....	20
III.2.1. Lớp Account và các lớp kế thừa.....	20
III.2.1.1. Lớp Member và các lớp Buyer, Seller và Staff	21

III.2.1.2. Lớp Admin	23
III.2.2. Lớp Product	24
III.2.3. Lớp Order	24
III.2.4. Lớp Voucher	25
III.3. Các lớp “Provider”	26
III.3.1. Lớp LoginProvider	27
III.3.2. Lớp AccountProvider	28
III.3.3. Lớp OrderProvider	29
III.3.4. Lớp ProductProvider	30
III.3.5. Lớp VoucherProvider	31
III.4. Các lớp giao diện.....	31
III.4.1. Lớp MyDialog và các lớp kế thừa.....	32
III.4.1.1. Lớp FlexDialog và các lớp ProductDialog, VoucherDialog, EventDialog	32
III.4.1.2 Lớp RateOrderDialog và OrderDetailDialog	33
III.4.1.4. Lớp CheckoutDialog và CartEditDialog	34
III.4.1.5. Lớp StaffInforDialog và AddStaffDialog	34
III.4.2. Lớp MainWindow	34
III.5. Mối quan hệ giữa các lớp	35
III.5.1. Giữa các lớp Provider	35
III.5.1.1. AccountProvider và LoginProvider, ProductProvider, VoucherProvider	35
III.5.1.2. ProductProvider và OrderProvider, VoucherProvider	37
III.5.2. Giữa các lớp Provider và các lớp thành phần.....	37
III.5.2.1. Lớp AccountProvider và lớp Admin	37
III.5.2.2. Lớp ProductProvider và các lớp Seller, Buyer.....	38
III.5.2.3. Lớp OrderProvider và các lớp Product, Admin, Member, Seller	39
III.5.2.4. Lớp VoucherProvider và các lớp Product, Seller.....	42
III.5.3. Giữa các lớp Provider và các lớp giao diện.....	43
III.5.3.1. ProductProvider và ProductDialog	43
III.5.3.2. OrderProvider và CheckoutDialog	44

III.5.3.3. VoucherProvider và VoucherDialog, EventDialog.....	44
III.5.4. Giữa các lớp thành phần	45
III.5.5. Giữa các lớp thành phần và các lớp giao diện.....	45
IV. Môi trường phát triển	47
V. Tổng kết	48
VI. Tự đánh giá	49
VI.1. Bảng điểm	49
VI.2. Đóng góp của các thành viên.....	49
VII. Tài liệu tham khảo.....	50

I. Giới thiệu

Bily Flower Shop là cửa hàng hoa tươi (giả định) chuyên cung cấp các loại hoa tươi có nguồn gốc nội, ngoại nhập cùng các dịch vụ liên quan đến hoa với chất lượng luôn được đánh giá cao.



Với phương châm muốn đem đến cho khách hàng những trải nghiệm tốt nhất, đồng thời hướng đến mục tiêu phát triển thương hiệu lâu dài và tìm kiếm những khách hàng tiềm năng, **Bily** nhận thấy sự cần thiết phải mở rộng quy mô kinh doanh và xây dựng thương hiệu vững chắc thông qua mạng Internet – một môi trường giàu tiềm năng cho những thương hiệu còn non trẻ. Đó là lý do ứng dụng **Bily Flower Shop** ra đời.

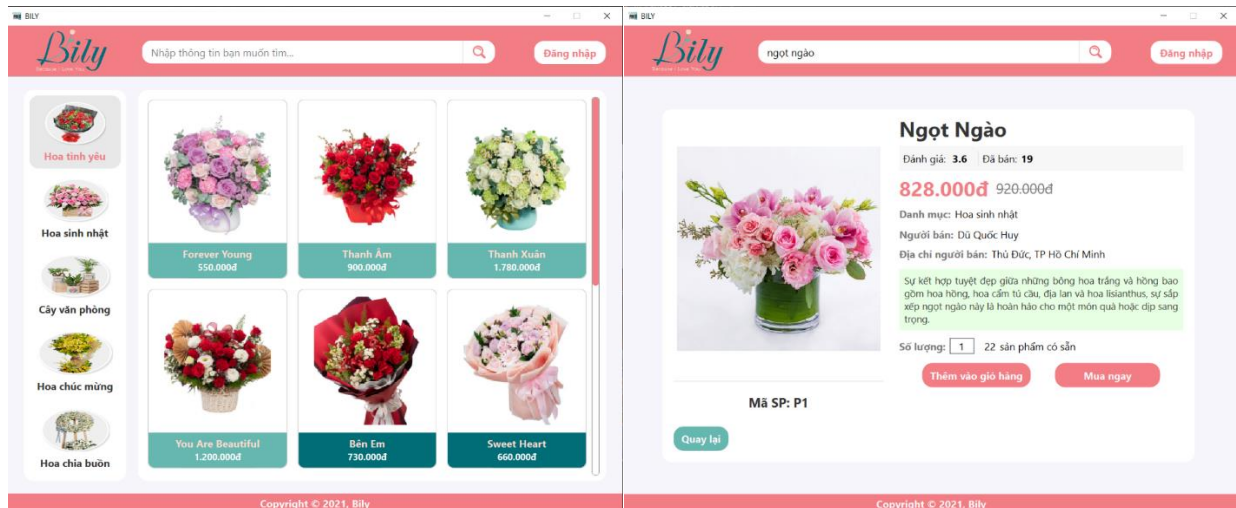
Bily Flower Shop là một “cộng đồng” thu nhỏ dành cho những người yêu hoa. Ai cũng có thể tạo tài khoản và tham gia vào cộng đồng ấy chỉ với một vài thao tác đăng ký đơn giản. Ứng dụng xây dựng đa dạng các thành phần người tham gia, từ đội ngũ quản trị viên (*admin*), nhân viên (*staff*) cho đến khách hàng (*buyer*) và những người có nhu cầu tìm đến **Bily** để tìm khách hàng cho họ (*seller*).

Với **Bily Flower Shop**, chúng em hi vọng những người bán hoa chạm đến gần hơn các khách hàng tiềm năng của mình, và những người mua hoa có thêm một kênh thông tin hữu ích, một ứng dụng uy tín sẵn sàng hỗ trợ họ trong các dịch vụ liên quan. Đồng thời, đây sẽ là bước đệm giúp **Bily** xây dựng uy tín và có được số lượng khách hàng ổn định hơn.

II. Các chức năng cơ bản

II.1. Xem hoa theo danh mục, tìm kiếm hoa

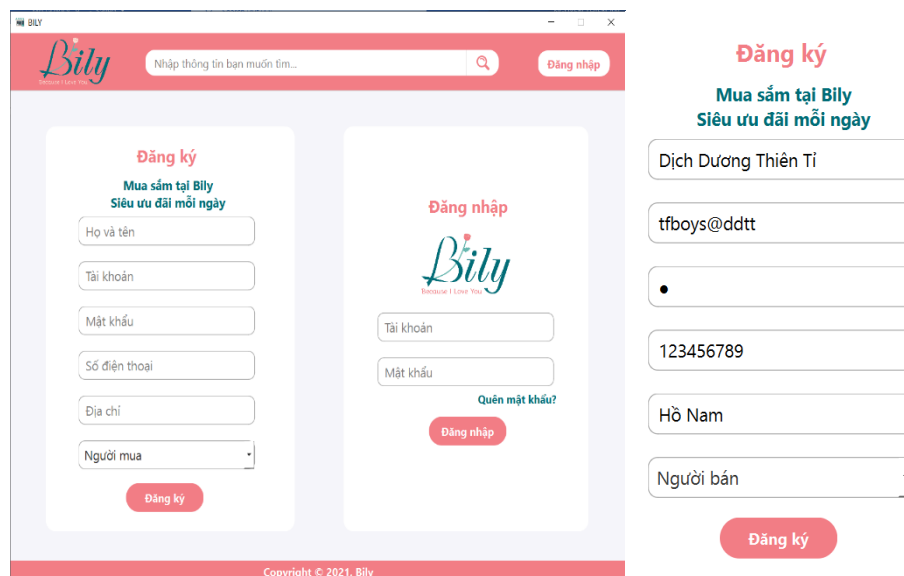
Ứng dụng **Bily Flower Shop** chia các loại hoa thành 5 danh mục, bao gồm: Hoa tình yêu, hoa sinh nhật, cây văn phòng, hoa chúc mừng, hoa chia buồn. Bên cạnh đó, ứng dụng có hỗ trợ tính năng tìm kiếm hoa theo từ khóa nhằm giúp người dùng có thể dễ dàng tìm được loại hoa theo nhu cầu.



Hình II.1.1. Xem hoa theo danh mục và tìm kiếm hoa

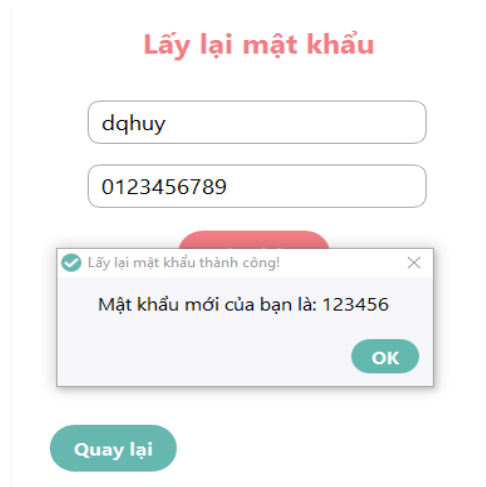
II.2. Đăng ký, đăng nhập

Người dùng có thể đăng ký 2 loại tài khoản là *người mua* hoặc *người bán*:



Hình II.2.1. Chức năng đăng ký, đăng nhập

Trong trường hợp quên mật khẩu, hệ thống có hỗ trợ người dùng đặt lại mật khẩu mới (mặc định là **123456**), với điều kiện là họ cung cấp chính xác *tên đăng nhập* và *số điện thoại* ứng với tài khoản đó.



Hình II.2.2. Chức năng tìm lại mật khẩu

II.3. Hệ thống tài khoản

Có tất cả 4 loại tài khoản (tương ứng với 4 vai trò) trong **Bily Flower Shop**, bao gồm:

- Quản lý (*admin*)
- Nhân viên (*staff*)
- Người mua (*buyer*)
- Người bán (*seller*)

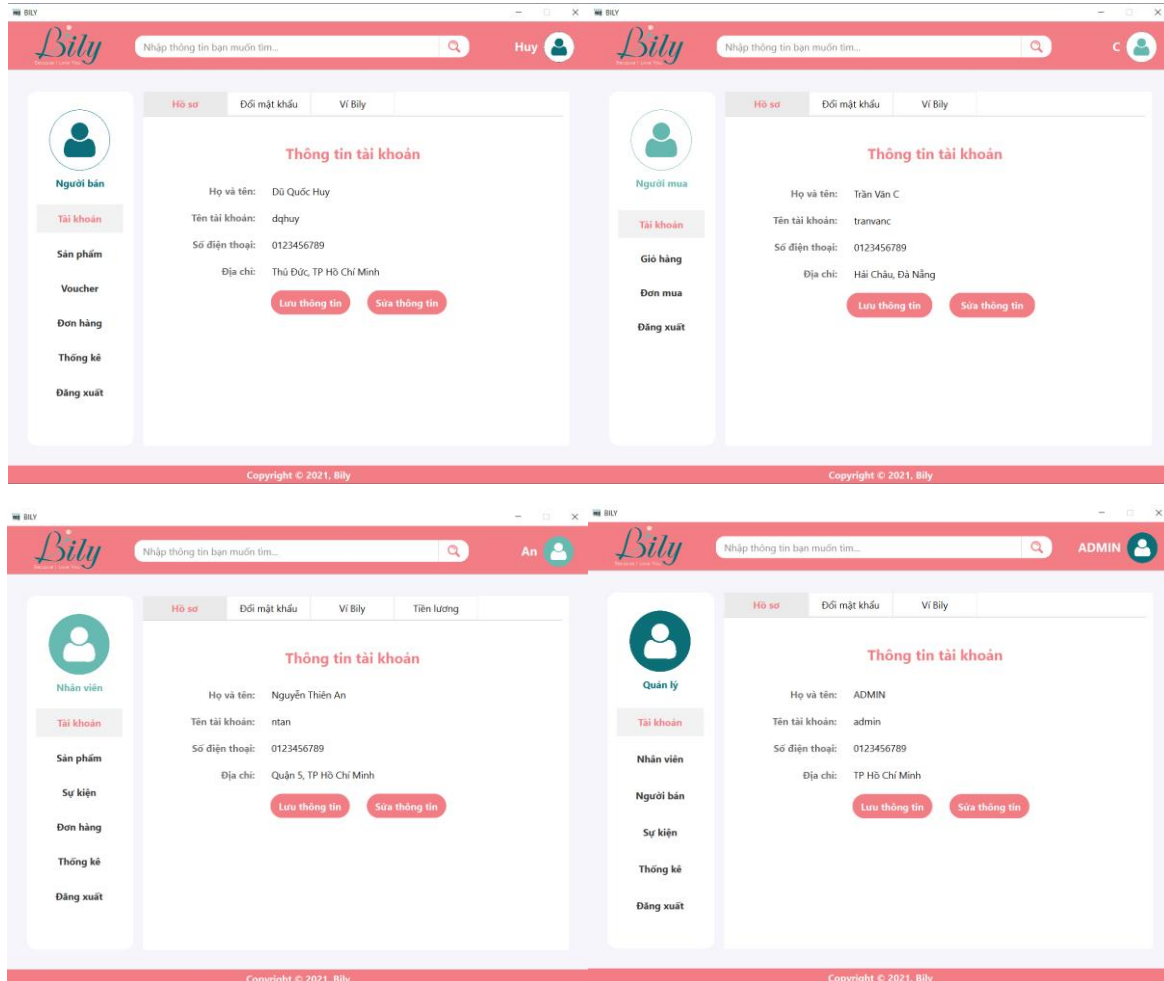


Hình II.3.1. Ảnh đại diện của các loại tài khoản

Trong đó, *quản lý* chịu trách nhiệm quản lý và điều hành các hoạt động diễn ra trên ứng dụng; *nhân viên* tổ chức các sự kiện giảm giá sản phẩm và bán những sản phẩm của cửa hàng; *người bán* đăng và bán các sản phẩm về hoa; *người mua* tìm kiếm cho mình những sản phẩm yêu thích và chốt đơn.

II.3.1. Trang quản lý tài khoản

Mỗi loại tài khoản tương ứng sẽ có trang quản lý tài khoản khác nhau. Để truy cập vào trang quản lý tài khoản của mình, người dùng có thể nhấn vào phần ảnh đại diện ở góc trên bên phải của cửa sổ.



Hình II.3.2. Trang quản lý tài khoản của các loại tài khoản

Người dùng có thể thay đổi thông tin cá nhân, thay đổi mật khẩu hoặc truy cập vào **Ví điện tử Bily** thông qua các tab ở phần **Tài khoản**.

II.3.2. Ví điện tử Bily

Các tài khoản trong hệ thống sử dụng ví điện tử **Bily** (giả định) để thực hiện thanh toán hoặc trả lương. Ngoài ra người dùng có thể nạp tiền hoặc rút tiền với vài bước đơn giản.

Ví Bily

Số dư hiện tại: **150.000đ**

Nạp tiền:

Nạp ngay

Rút tiền:

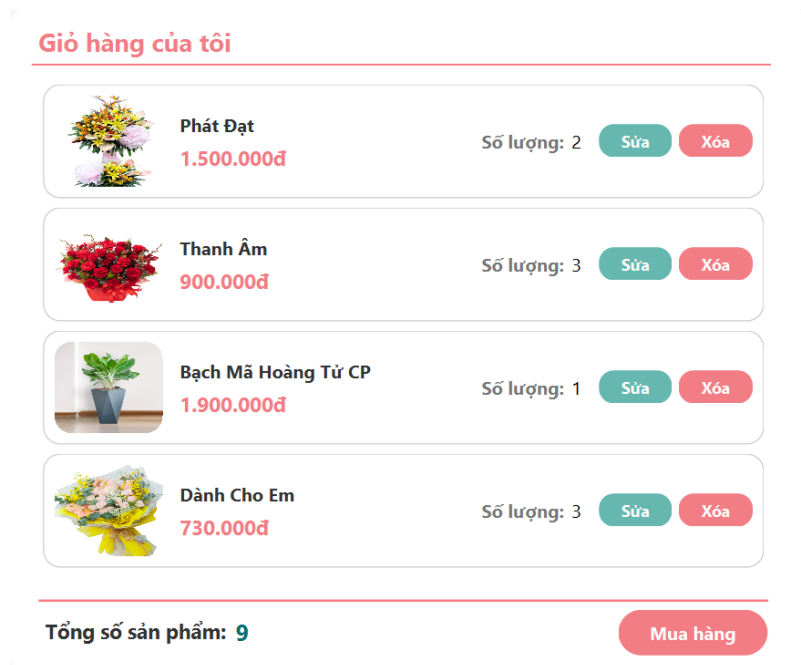
Rút tiền

Hình II.3.3. Ví Bily

II.3.3. Người mua

II.3.3.1. Sử dụng giỏ hàng

Người mua có thể chọn ra sản phẩm mình yêu thích và đưa nó vào giỏ hàng của họ. Sau khi thêm, ở mục **“Giỏ hàng”** sẽ hiển thị theo danh sách các sản phẩm đã có trong giỏ cùng với số lượng và giá tiền kèm theo.



Hình II.3.4. Giỏ hàng của người mua

Tương tự, người mua hoàn toàn có thể xóa sản phẩm khỏi giỏ hàng của mình khi không còn nhu cầu. Người dùng nhấn nút **“Mua hàng”** để chốt đơn đặt hàng tất cả sản phẩm đang có trong giỏ.

II.3.3.2. Đặt hàng, hủy đơn hàng

Sau khi người dùng chọn **“Mua hàng”**, hộp thoại thanh toán sẽ hiện ra để người dùng cung cấp những thông tin như địa chỉ, ghi chú giao hàng và chọn hình thức thanh toán (*thanh toán khi nhận hàng (COD)* hoặc dùng *Ví Bily*).

Thanh toán

Thanh toán

Địa chỉ giao hàng:
Lam Sơn, Thanh Hóa

You Are Beautiful
1.200.000đ

Số lượng: 1

Ghi chú giao hàng:
Chúc mừng sinh nhật honeyyy <3

Sử dụng Ví Bily

Tổng tiền: 1.200.000đ
Thanh toán

Hình II.3.5. Cung cấp thông tin khi thanh toán

Sau khi chọn thanh toán, những đơn hàng sẽ được tạo ra và đưa vào danh sách đơn hàng đang chờ duyệt của người mua này với người bán tương ứng.

Đã được xác nhận	Chưa được xác nhận	Đã được xác nhận	Chưa được xác nhận
<div> <div> Trầu Bà XM Số lượng: 2 Tổng tiền: 330.000đ </div> <div> Mã đơn hàng: O4 07/12/2021 Đánh giá </div> </div>		<div> <div> Khai Trương Hồng Phát Số lượng: 1 Tổng tiền: 1.000.000đ </div> <div> Mã đơn hàng: O5 17/12/2021 Hủy </div> </div>	
<div> <div> Đò Cổ Điển Số lượng: 1 Tổng tiền: 1.780.000đ </div> <div> Mã đơn hàng: O19 06/01/2022 Đánh giá </div> </div>		<div> <div> It's you Số lượng: 1 Tổng tiền: 1.500.000đ </div> <div> Mã đơn hàng: O14 06/01/2022 Hủy </div> </div>	
<div> <div> Ngọc Ngân CPS Số lượng: 1 Tổng tiền: 1.750.000đ </div> <div> Mã đơn hàng: O15 06/01/2022 Đánh giá </div> </div>		<div> <div> Hưng Thịnh Số lượng: 1 Tổng tiền: 1.150.000đ </div> <div> Mã đơn hàng: O17 06/01/2022 Hủy </div> </div>	
<div> <div> Lệ Buồn Số lượng: 1 Tổng tiền: 1.500.000đ </div> <div> Mã đơn hàng: O16 06/01/2022 Đánh giá </div> </div>		<div> <div> Bên Em Số lượng: 1 Tổng tiền: 730.000đ </div> <div> Mã đơn hàng: O18 06/01/2022 Hủy </div> </div>	

Hình II.3.6. Danh sách đơn hàng của người mua

Người mua có thể hủy những đơn hàng chưa được người bán duyệt. Với những đơn hàng đã được duyệt, sản phẩm sẽ được gửi đến cho người mua trong vài ngày và sau khi nhận hàng, người mua có thể gửi đánh giá cho chất lượng dịch vụ của người bán đó (đánh giá từ 1 – 5 sao).

Hình II.3.7. Chức năng đánh giá chất lượng dịch vụ

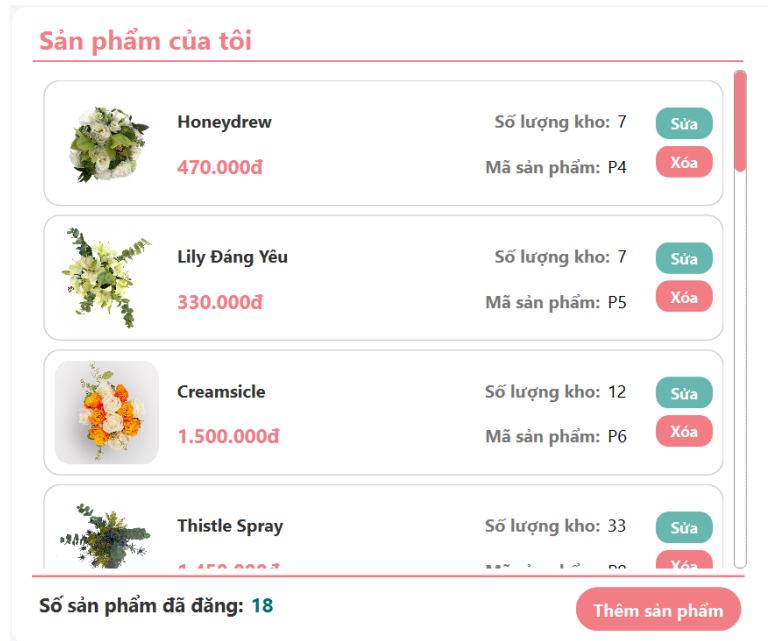
II.3.4. Người bán

II.3.4.1. Quản lý gian hàng

Người bán có thể thêm hoặc chỉnh sửa những hoa trong danh sách hoa đang mà họ đang bán bằng cách cung cấp những thông tin cần thiết cho sản phẩm.

Hình II.3.8. Chức năng thêm, chỉnh sửa sản phẩm

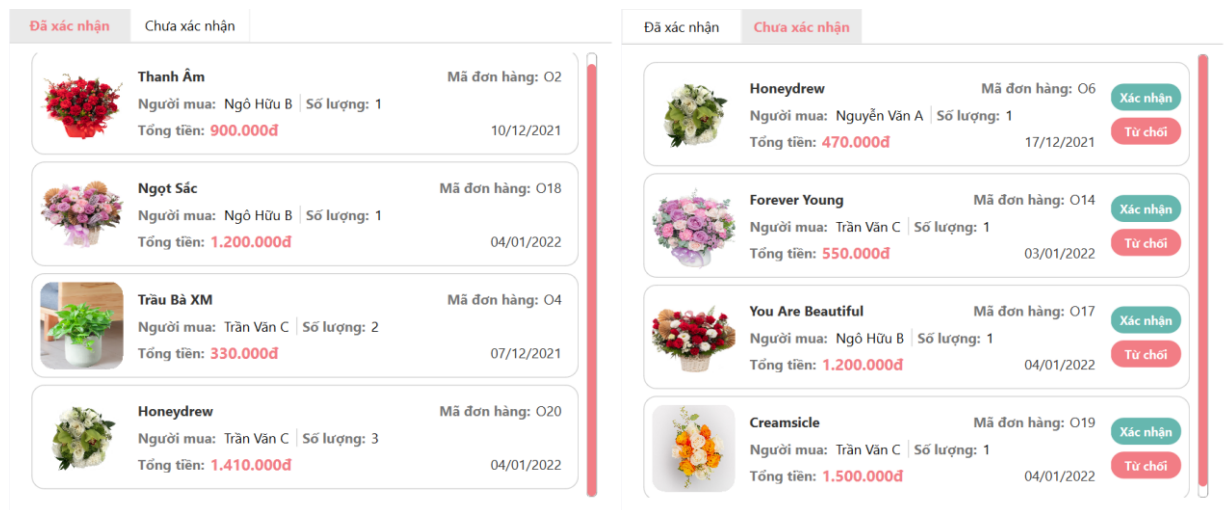
Người bán cũng có thể xem danh sách những sản phẩm đã đăng và xóa một sản phẩm nào đó.



Hình II.3.9. Xem danh sách sản phẩm đã đăng

II.3.4.2. Quản lý các đơn hàng

Người bán có thể xem chi tiết đơn hàng nhận được, chấp nhận hoặc từ chối đơn hàng đó. Khi chấp nhận đơn hàng, người bán sẽ nhận được một khoản thu bằng 95% giá trị đơn hàng (vì hệ thống thu phí 5% trên mỗi đơn hàng).



Hình II.3.10. Chức năng quản lý các đơn hàng của người bán

II.3.4.3. Quản lý voucher

Người bán có thể tạo các voucher áp dụng cho các sản phẩm nhất định của họ. Mỗi voucher sẽ mang một số thông tin cơ bản như tên, phần trăm giảm giá và những sản phẩm được áp dụng. Ngoài ra, họ cũng có thể chỉnh sửa hoặc xóa các voucher đã đăng.

Voucher

Tạo Voucher

Tên voucher:

Sale tới 40% dọn kho đón Tết

Phần trăm giảm:

40

Sản phẩm áp dụng:

☐ Tất cả
☒ P4 - Honeydrew
☒ P5 - Lily Đáng Yêu
☒ P6 - Creamsicle
☐ P9 - Thistle Spray
☐ P18 - Tiến Bước Thuận Lợi

Lưu

Hình II.3.11. Chức năng tạo voucher

Quản lý Voucher

Giảm giá ngẫu nhiên

Phần trăm giảm: 50

Mã Voucher: V2

Sửa

Xóa

Sale tới 40% dọn kho đón Tết

Phần trăm giảm: 40

Mã Voucher: V4

Sửa

Xóa

Sale đậm sâu nè cả nhà

Phần trăm giảm: 40

Mã Voucher: V5

Sửa

Xóa

Hoa cười người héo vì Bily sale nhiều quáaaa

Phần trăm giảm: 25

Mã Voucher: V6

Sửa

Xóa

Số Voucher đã tạo: 4

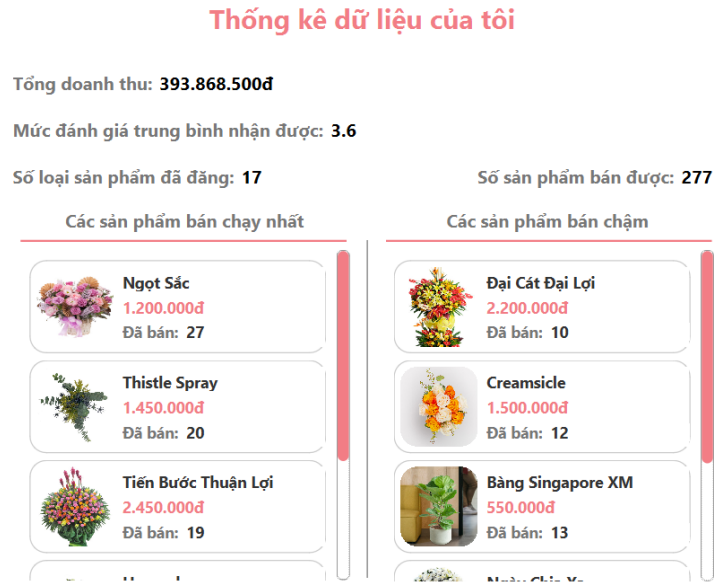
Tạo Voucher

Hình II.3.12. Danh sách các voucher của người bán

II.3.4.4. Thống kê dữ liệu

Người bán có thể xem được những thông tin như sau:

- Tổng doanh thu
- Tổng số hoa đã bán được
- Mức đánh giá trung bình nhận được
- Những hoa bán chạy nhất và bán chậm



Hình II.3.13. Thống kê dữ liệu của người bán

II.3.5. Nhân viên

Tài khoản của các *nhân viên* cũng có thể thực hiện những chức năng liên quan đến sản phẩm, đơn hàng và thống kê mà tài khoản của *người bán* có thể thực hiện. Ngoài ra, mỗi *nhân viên* còn có thêm tiền lương, tiền thưởng trong tháng hiện tại cùng và tạo ra sự kiện cho cửa hàng.

II.3.5.1. Tiền lương và tiền thưởng

Nhân viên sẽ có lương cơ bản và tiền thưởng. Với mỗi sản phẩm bán được, *nhân viên* được thưởng một khoản bằng 10% giá trị sản phẩm đó.

Hồ sơ	Đổi mật khẩu	Ví Bily	Tiền lương
Mức lương hiện tại			
Lương cơ bản:		20.000.000đ	
Thưởng thêm (tháng này):		500.000đ	
Tổng cộng:		20.500.000đ	

Hình II.3.14. Trang thông tin về tiền lương của nhân viên

II.3.5.2. Quản lý sự kiện

Những *voucher* mà *nhân viên* tạo ra sẽ áp dụng cho tất cả sản phẩm trong hệ thống của **Bily**, ta gọi đó là các sự kiện (*event*). Nếu một sản phẩm đang được giảm giá theo nhiều mức giá khác nhau, mức giá thấp nhất sẽ được áp dụng cho người mua.

Quản lý Sự kiện

Sale tối cận không kịp nè cả nhà
Phân trăm giảm: 15

Mã sự kiện: V4

SửaXóa

Sự kiện

Tạo sự kiện

Tên sự kiện

Hoa về đầu năm - Tài lộc cả năm

Phân trăm giảm:

20

Lưu

Số Sự kiện đã tạo: 1

Tạo Sự kiện

Hình II.3.15. Chức năng quản lý sự kiện của nhân viên

II.3.6. Quản lý của cửa hàng

II.3.6.1. Quản lý các nhân viên

Quản lý có thể thêm một người bán nào đó thành nhân viên của cửa hàng hoặc “sa thải” một nhân viên. Tài khoản của nhân viên bị sa thải sẽ hoàn toàn bị xóa ra khỏi dữ liệu của hệ thống.

Danh sách nhân viên

ntan Nguyễn Thiên An Lương tháng này: 20.500.000đ

Sa thải

htvu

Quản lý nhân viên

Sa thải

dqhuý

Sa thải

Thêm nhân viên

Chọn các Người bán được thêm làm nhân viên:

☒ tfboys@ddtt - Dịch Dương Thiên Ti

Lưu

Số lượng nhân viên: 1

Trả lương

Thêm nhân viên

Danh sách nhân viên

ntan Nguyễn Thiên An Lương tháng này: 20.500.000đ

Sa thải

demoS Troioiii Lương tháng này: 7.000.000đ

Sa thải

htvu He

Xác nhận thao tác

Bạn có chắc chắn muốn sa thải nhân viên Troioiii?

Có Không

Sa thải

dqhuý D

Sa thải

Số lượng nhân viên: 4

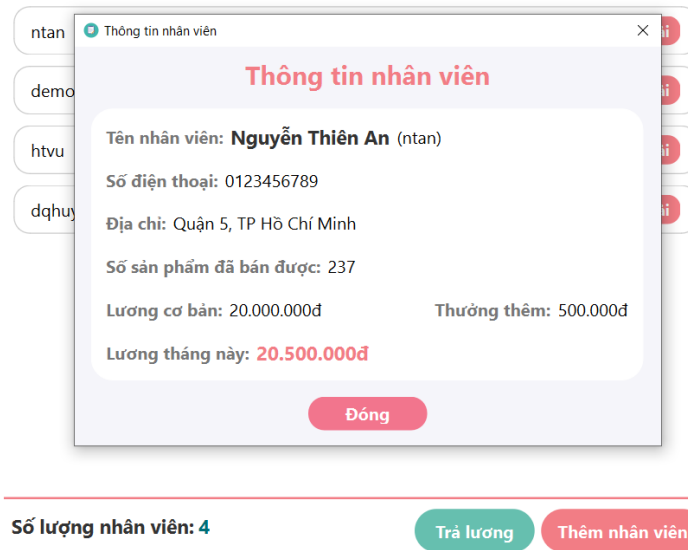
Trả lương

Thêm nhân viên

Hình II.3.16. Chức năng thêm, sa thải nhân viên

Quản lý có thể xem thông tin về một nhân viên nào đó.

Danh sách nhân viên

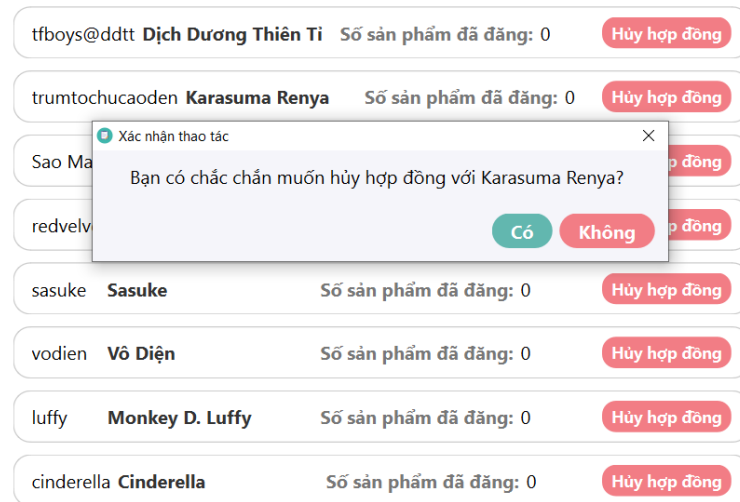


Hình II.3.17. Chức năng xem thông tin nhân viên

II.3.6.2. Quản lý hợp đồng người bán

Quản lý có thể chấm dứt hợp đồng với người bán bất kỳ nếu nhận thấy người bán đó có những sai phạm với chính sách của hệ thống. Tài khoản của người bán bị hủy hợp đồng sẽ hoàn toàn bị xóa khỏi dữ liệu.

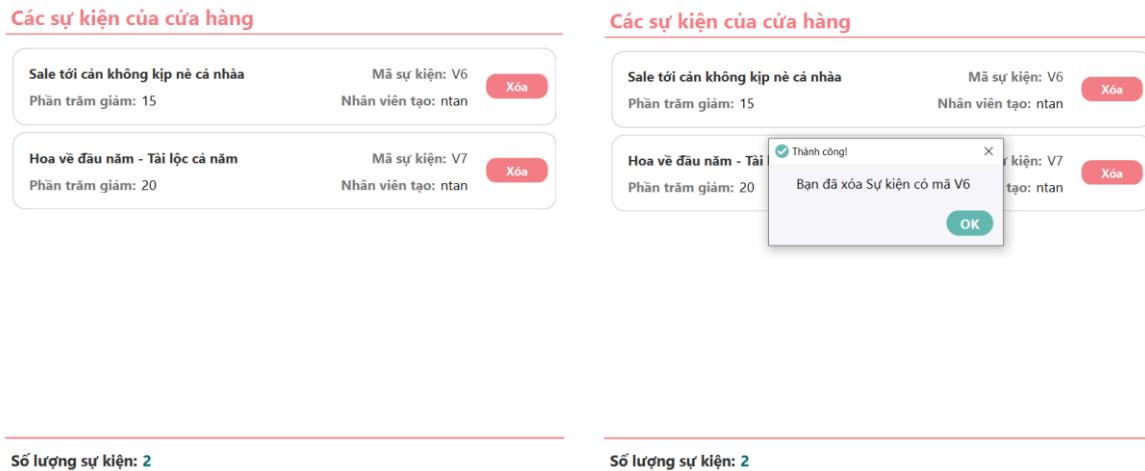
Danh sách người bán



Hình II.3.18. Chức năng chấm dứt hợp đồng

II.3.6.3. Quản lý các sự kiện trong cửa hàng

Quản lý có thể xóa một sự kiện bất kì do nhân viên nào đó đăng mà người này cảm thấy không hợp lý.



Hình II.3.19. Chức năng quản lý các sự kiện của cửa hàng

III. Thiết kế đồ án

III.1. Tổ chức dữ liệu

Dữ liệu của ứng dụng được lưu trong các file có định dạng JSON và TXT. Trong đó:

- Các file TXT lưu dữ liệu hỗ trợ cho thuật toán tìm kiếm sản phẩm.
- Các file JSON lưu thông tin các tài khoản, sản phẩm, đơn hàng và vouchers.

Để có thể thực hiện việc tìm kiếm, thông tin chính của các sản phẩm (được rút ra từ tên và mô tả của nó) sẽ được lưu trữ dưới dạng chữ không dấu.

```
P15
khai truong hong phat ke hoa tone mau hong hien dai tre trung noi bat
ke hoa truyen thong nghia chuc tai loc hop tang dip chuc mung khai
truong ki niem
```

Hình III.1.1. Minh họa cho thông tin chính của sản phẩm có ID P15

Hệ thống sử dụng tổng cộng 5 file JSON, bao gồm:

- **passwords.json**: Lưu tên đăng nhập và mật khẩu (đã qua bước mã hóa SHA256)
- **accounts.json**: Lưu thông tin của các tài khoản
- **products.json**: Lưu thông tin của các sản phẩm
- **vouchers.json**: Lưu thông tin của các voucher
- **orders.json**: Lưu thông tin của các đơn hàng

Đồng thời, thư viện được sử dụng để thao tác với các file JSON là **JSON for Modern C++**.

```
"INFOS": {
  "admin": {
    "address": "TP Hồ Chí Minh",
    "balance": 6073693499,
    "name": "ADMIN",
    "phoneNumber": "0123456789",
    "role": "ADMIN"
  },
  "demos": {
    "address": "1",
    "balance": 0,
    "name": "1",
    "phoneNumber": "1",
    "role": "S"
  },
  "dqhuy": {
    "address": "Thủ Đức, TP Hồ Chí Minh",
    "balance": 150000,
    "name": "Dũ Quốc Huy",
    "phoneNumber": "0123456789",
    "role": "S"
  }
},
"INFOS": {
  "P1": {
    "category": 1,
    "description": "Sự kết hợp tuyệt đẹp giữa những bông hoa trắng và hồng bao gồm hoa hồng, hoa cẩm tú cầu, địa lan và hoa lisianthus, sự sắp xếp ngọt ngào này là hoàn hảo cho một món quà hoặc dịp sang trọng.",
    "discounts": [[10.0, 1]],
    "imageDir": ":/images/products/birthday/birthday1.jpg",
    "name": "Ngọt Ngào",
    "originalPrice": 920000,
    "ratings": [1, 3, 3, 7, 4],
    "salePrice": 828000,
    "sellerUsername": "dqhuy",
    "sold": 19,
    "stock": 22
  },
  "P10": {
    "category": 1,
    "description": "Bó hoa này tỏa ra sự ấm áp trong những bông hoa tone vàng như hoa hồng cam, hoa cẩm chướng hồng đào.",
    "discounts": [],
    "imageDir": ":/images/products/birthday/birthday10.jpg",
    "name": "Ánh Bình Minh",
    "originalPrice": 600000,
    "ratings": [1, 2, 6, 1, 2],
```

Hình III.1.2. Minh họa dữ liệu lưu trong các file accounts.json và products.json

III.2. Các lớp thành phần

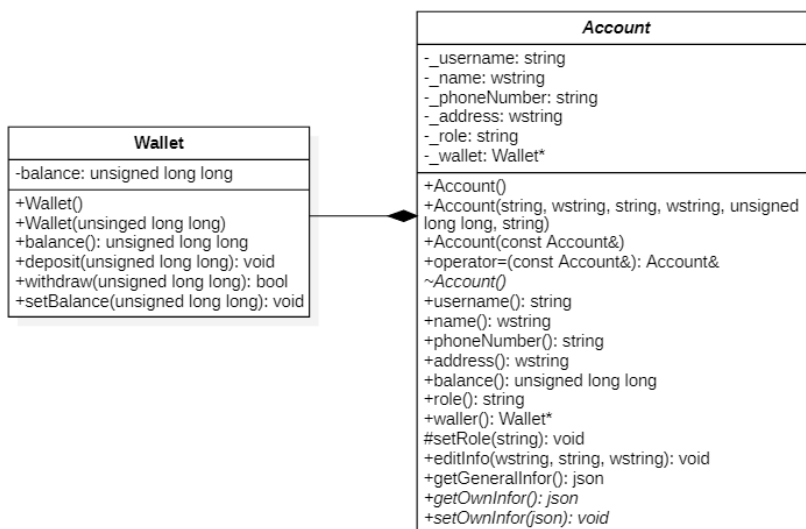
Trong phần này, ta sẽ đề cập sơ lược đến những lớp thành phần trong hệ thống. Những tương tác qua lại giữa chúng sẽ được trình bày trong phần **III.5. Mối quan hệ giữa các lớp**.

Lưu ý rằng, những lớp thành phần đều có sử dụng đến lớp **json** từ thư viện *JSON for Modern C++* trong các thao tác trả về thông tin dưới dạng JSON (tức là trả về một đối tượng thuộc lớp **json**) và trích xuất thông tin từ một đối tượng thuộc lớp **json**.

III.2.1. Lớp Account và các lớp kế thừa

Mọi tài khoản trong hệ thống (ứng với một đối tượng thuộc lớp **Account**) đều có những thông tin chung là *tên đăng nhập, họ và tên, số điện thoại, địa chỉ, vai trò* và một *ví điện tử* riêng với thông tin cơ bản là *số dư của ví* (lớp **Wallet**, tương ứng với chức năng *Ví Bily*). Tất cả đều có thể thực hiện một số thao tác cơ bản lấy thông tin chung, thay đổi thông tin và nạp tiền, rút tiền. Hệ thống sẽ định danh mỗi tài khoản thông qua tên đăng nhập. Mối quan hệ giữa lớp **Account** và **Wallet** là *quan hệ bao hàm phụ thuộc (composition)*.

Ngoài ra, mỗi loại tài khoản sẽ có một số thông tin đặc trưng khác. Lớp **Account** sẽ có hai *phương thức thuần ảo* cho những thao tác trả về thông tin đặc trưng với dưới dạng JSON và trích xuất thông tin từ một đối tượng thuộc lớp **json**. Do đó, lớp **Account** là một *lớp trừu tượng*.



Hình III.2.1. Sơ đồ lớp của lớp Account và Wallet

Những lớp kế thừa trực tiếp từ lớp **Account** là lớp **Admin** và lớp **Member**. Trong đó, tài khoản là đối tượng thuộc lớp **Admin** chính là quản lý *duy nhất* của hệ thống, tất cả những tài khoản còn lại đều là đối tượng thuộc lớp **Member**.

III.2.1.1. Lớp Member và các lớp Buyer, Seller và Staff

Bên cạnh các thông tin chung của mọi tài khoản, những tài khoản, hay là đối tượng, thuộc lớp **Member** có thêm *danh sách ID những đơn hàng đang chờ duyệt và những đơn hàng đã được duyệt*.

Những thông tin cần được xử lý khi chấp nhận hoặc từ chối/hủy đơn hàng sẽ có sự khác nhau giữa các tài khoản, tùy vào vai trò của tài khoản đó. Chính vì thế, các phương thức *addOrder*, *acceptOrder* và *removeOrder* trong lớp **Member** được chỉ định là *phương thức ảo*. Tuy nhiên, những thao tác chung luôn được thực hiện trong các phương thức này là điều chỉnh danh sách những đơn hàng đang chờ duyệt và đã được duyệt. Ngoài ra, lớp **Member** cũng có hai *phương thức ảo* cho các thao tác trả về thông tin đặc trưng với dưới dạng JSON và trích xuất thông tin từ một đối tượng thuộc lớp **json** (ghi đè từ lớp **Account**).

Member
#_acceptedOrders: list<string> #_pendingOrders: list<string>
+Member() +Member(string, wstring, string, wstring, unsigned long long, string) ~Member() +acceptedOrders(): list<string> +pendingOrders(): list<string> +getOwnInfor(): json +setOwnInfor(json): void +addOrder(const string&): void +acceptOrder(const string&): void +removeOrder(const string&, bool): void

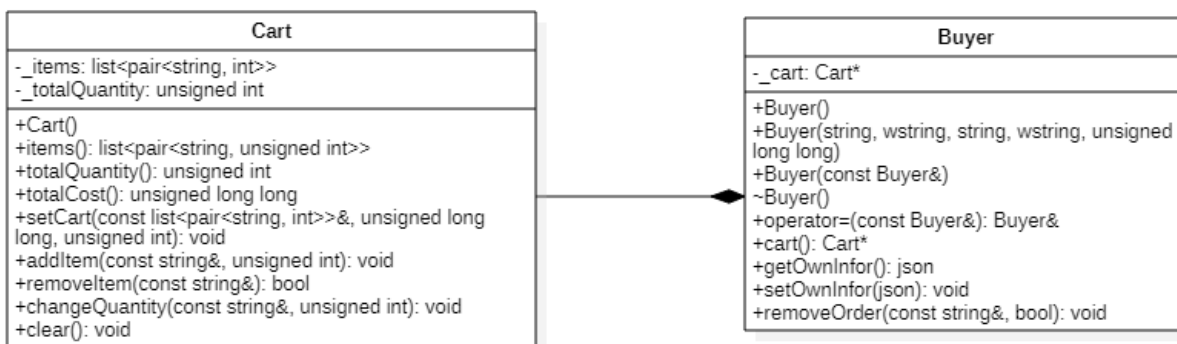
Hình III.2.2. Sơ đồ lớp Member

Những lớp kế thừa trực tiếp từ lớp **Member** là lớp **Buyer**, **Seller**.

III.2.1.1.1. Lớp Buyer

Tài khoản của các người mua trong hệ thống đều là đối tượng thuộc lớp **Buyer**. Ngoài những thuộc tính kế thừa được từ lớp **Account** và **Member**, mỗi đối tượng thuộc lớp **Buyer** có thêm cho riêng mình một *giỏ hàng* (lớp **Cart**). *Giỏ hàng* sẽ lưu những thông tin như các sản phẩm đang có trong giỏ và số lượng mỗi sản phẩm. Mối quan hệ giữa lớp **Buyer** và lớp **Cart** là quan hệ bao hàm phụ thuộc (*composition*).

Bên cạnh đó, lớp **Buyer** sẽ có 2 phương thức ghi đè lại các *phương thức ảo* từ lớp **Member**, chính là những phương thức liên quan đến thông tin đặc trưng và dữ liệu dạng JSON.



Hình III.2.3. Sơ đồ lớp Buyer và Cart

Những tài khoản có vai trò là *người mua* đều có thể thêm, loại sản phẩm khỏi giỏ hàng, thay đổi số lượng các sản phẩm đã thêm trong giỏ hàng của họ. Phương thức *removeOrder*

trong lớp **Buyer** ghi đè từ *phương thức ảo* trong lớp **Member**. Việc thanh toán những sản phẩm trong giỏ hàng và hủy đơn hàng của đối tượng thuộc lớp **Buyer** sẽ cần tự tương tác với một số lớp khác.

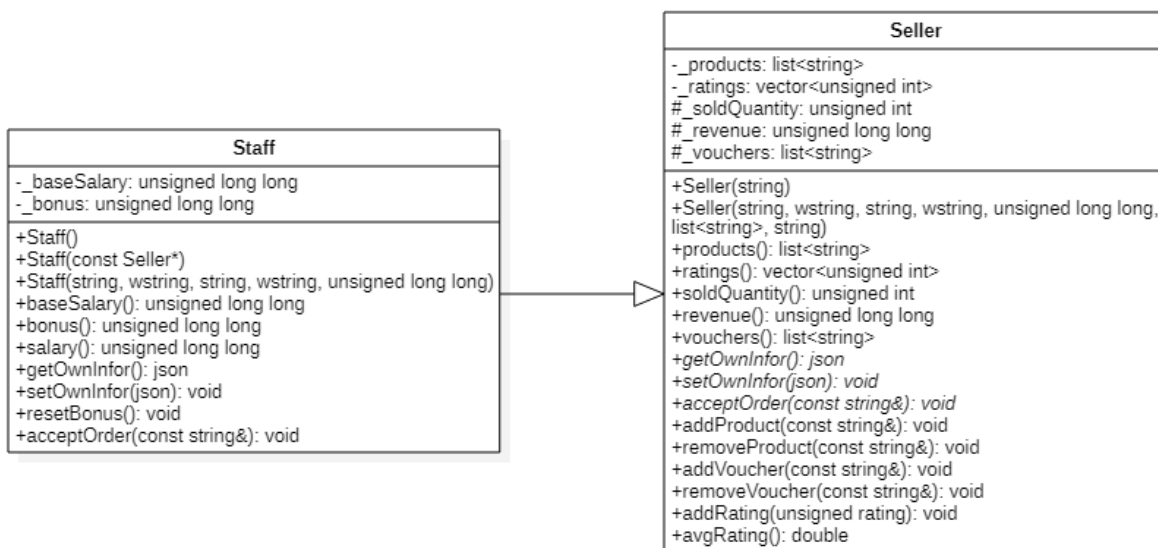
III.2.1.1.2. Lớp Seller và Staff

Các tài khoản của những *người bán*, hay đối tượng thuộc lớp **Seller**, đều có chứa những thông tin đặc trưng sau:

- Danh sách ID của các sản phẩm đã đăng
- Các đánh giá đã nhận được từ người mua
- Số lượng sản phẩm đã bán
- Doanh thu
- Danh sách ID của các voucher mà họ đã tạo

Những thao tác cơ bản như thêm và xóa sản phẩm, tạo và xóa các *voucher* đều có thể được thực hiện bởi *người bán*. Bên cạnh đó, phương thức *acceptOrder* trong lớp **Seller** ghi đè lại từ *phương thức ảo* trong lớp **Member**, và nó vẫn tiếp tục là *phương thức ảo*. Ngoài ra, lớp **Seller** ghi đè 2 phương thức liên quan đến thông tin đặc trưng và dữ liệu dạng JSON từ lớp **Member** (*getOwnInfor* và *setOwnInfor*). Tại đây, 2 phương thức đó tiếp tục là *phương thức ảo*.

Lớp **Staff** kế thừa trực tiếp từ lớp **Seller**. Theo đó, những *nhân viên* của cửa hàng cũng chính là *người bán*, chỉ khác ở điểm những sản phẩm họ đăng là sản phẩm của chính cửa hàng, và *voucher* mà họ tạo sẽ được xem là một *event*, áp dụng cho toàn bộ sản phẩm trong cửa hàng.



Hình III.2.4. Sơ đồ lớp Seller và lớp Staff

Mỗi *nhân viên* sẽ có thêm thông tin là *lương cơ bản* và *tiền thưởng trong tháng hiện tại*. Do vậy, 2 phương thức *getOwnInfor* và *setOwnInfor* của lớp này được ghi đè từ lớp **Seller**. Ngoài ra, phương thức *acceptOrder* trong lớp **Staff** được ghi đè từ *phương thức ảo* trong lớp **Seller**.

Việc cài đặt những phương thức *acceptOrder* của lớp **Seller** và **Staff** sẽ có tương tác với lớp mà ta chuẩn bị đề cập đến là lớp **Admin**.

III.2.1.2. Lớp Admin

Dựa theo tiêu chí hệ thống chỉ có duy nhất 1 quản lý, hay là chỉ có duy nhất 1 đối tượng thuộc lớp **Admin** được tạo ra, lớp **Admin** được cài đặt theo *mẫu thiết kế Singleton*.

Ngoài những thông tin chung từ lớp **Account**, lớp **Admin** có thêm các thông tin sau:

- Danh sách các nhân viên: Lưu *tên đăng nhập* của họ
- Doanh thu của cửa hàng (từ việc bán sản phẩm của chính cửa hàng)
- Lợi nhuận từ những đơn hàng thuộc các sản phẩm không phải của cửa hàng

Admin
-_staffs: list<string> -_shopRevenue: unsigned long long -_profitFromSeller: long long
-Admin() -Admin(string, wstring, string, wstring, unsigned long long) +getInstance(string, wstring, string, wstring, unsigned long long, unsigned long long, long long): Admin* ~Admin() +staffs(): list<string> +shopRevenue(): unsigned long long +profitFromSeller(): long long +getOwnInfor(): json +setOwnInfor(json): void +paySalaries(): bool +addStaff(const string&): void +dismissStaff(const string&): void +addShopRevenue(unsigned long long): void +addProfitFromSeller(long long): void +totalSalaries(): unsigned long long

Hình III.2.5. Sơ đồ lớp Admin

Đối tượng thuộc lớp **Admin** có thể trả lương cho các nhân viên, thêm một người bán nào đó thành nhân viên và sa thải nhân viên (đồng nghĩa với việc xóa tài khoản đó khỏi hệ thống). Những thao tác đó cần có sự tương tác giữa lớp **Admin** và một số lớp khác. Bên cạnh đó, lớp **Admin** có 2 phương thức ghi đè những *phương thức ảo* từ lớp **Account**, liên quan đến thông tin đặc trưng và dữ liệu dạng JSON.

III.2.2. Lớp Product

Lớp **Product** lưu thông tin chung của mỗi *sản phẩm* có trong hệ thống, bao gồm:

- ID sản phẩm
- Danh mục của sản phẩm
- Tên đăng nhập của người bán
- Giới thiệu đôi nét về sản phẩm đó
- Mức giá gốc và giá bán hiện tại
- Số lượng sản phẩm đã bán và số lượng có sẵn trong kho
- Các đánh giá nhận được từ người mua (từ 1 đến 5 sao)

Ngoài ra, lớp **Product** cung cấp một số phương thức như lấy các thông tin, trả về một đối tượng thuộc lớp **json** chứa toàn bộ thông tin, thay đổi các thông tin sản phẩm và cập nhật lại giá bán hiện tại (khi sản phẩm này được áp dụng cho một *voucher* hay *event* nào đó). Lúc đó, mức giá bán

của sản phẩm sẽ được tính toán lại dựa theo mức giảm giá cao nhất (tính theo phần trăm giảm của *voucher* hay *event* tương ứng). Để thuận tiện cho thao tác này, cấu trúc dữ liệu được sử dụng để lưu trữ các mức giảm giá áp dụng cho sản phẩm là **multiset**.

Product
<pre>-_id: string -_category: unsigned int -_name: wstring -_sellerUsername: string -_description: wstring -_originalPrice: unsigned long long -_salePrice: unsigned long long -_discounts: multiset<pair<float, int>> -_stock: unsigned int -_sold: unsigned int -_ratings: vector<unsigned int> -_imageDir: string +Product() +Product(string, unsigned int, wstring, string, wstring, unsigned long long, multiset<pair<float, int>>, unsigned int, unsigned int, vector<unsigned int>, string) +Product(unsigned int, wstring, string, wstring, unsigned long long, unsigned int, string) +ID(): string +category(): unsigned int +name(): wstring +sellerUsername(): string +description(): wstring +originalPrice(): unsigned long long +salePrice(): unsigned long long +discounts(): multiset<pair<float, int>> +stock(): unsigned int +sold(): unsigned int +ratings(): vector<unsigned int> +avgRating(): double +imageDir(): string +getFullInfo(): json +setID(const string&): void +editInfo(Product*): void +addRating(unsigned int): void +getDiscount(): float +isInEvent(): bool +addDiscount(float, int): void +removeDiscount(float, int): void +changeSoldQuantity(int): void +changeStockQuantity(int): void</pre>

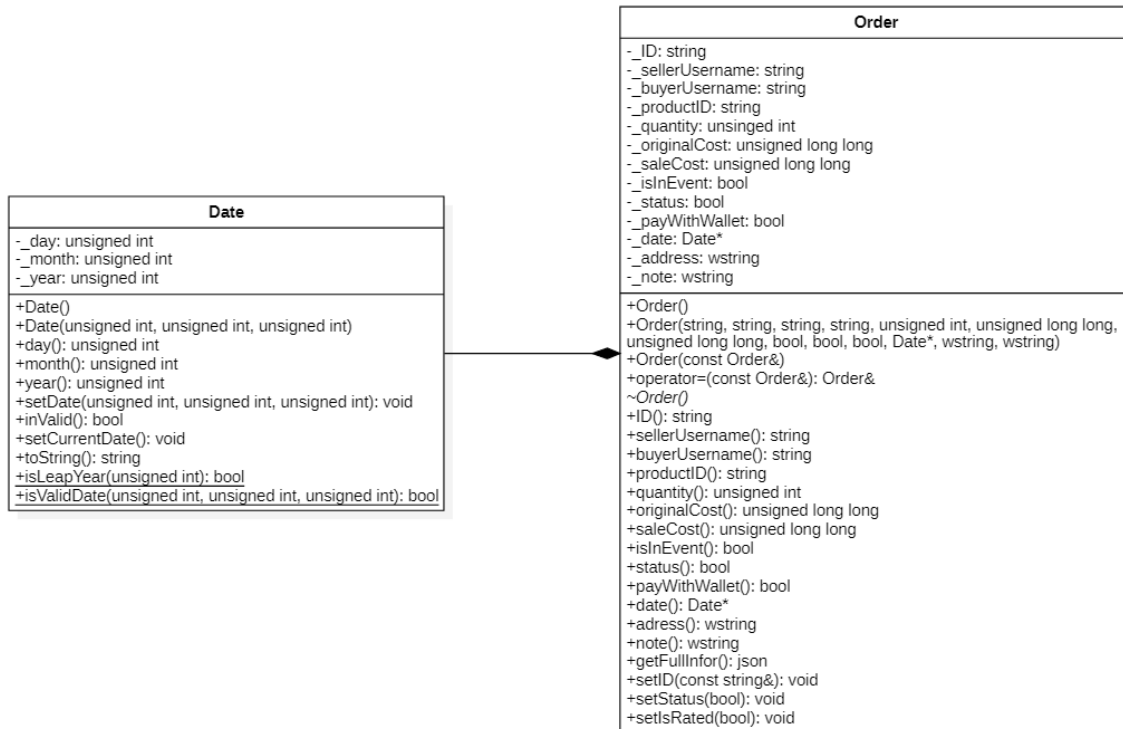
Hình III.2.6. Sơ đồ lớp Product

III.2.3. Lớp Order

Như tên gọi, lớp **Order** chứa các thông tin cần thiết của một đơn hàng cũng như các thao tác cơ bản như lấy thông tin, thay đổi trạng thái đơn hàng (đã duyệt, đã được đánh giá). Mỗi đối tượng thuộc lớp **Order** đều có các thuộc tính sau:

- ID đơn hàng
- Tên đăng nhập của *người bán* và *người mua*
- ID sản phẩm
- Số lượng sản phẩm đặt mua
- Giá gốc của sản phẩm
- Giá sản phẩm tại thời điểm đặt hàng
- Hình thức thanh toán: sử dụng **Ví Bily** hoặc thanh toán khi nhận hàng
- Ngày đặt hàng: một đối tượng thuộc lớp **Date**
- Địa chỉ giao hàng
- Ghi chú giao hàng

Đồng thời, để thuận tiện cho việc quản lý, ta cũng cần thêm các thuộc tính khác như kiểm tra đơn hàng đã được xác nhận chưa, đã được người mua đánh giá chưa, cũng như kiểm tra sản phẩm trong đơn có đang áp dụng *voucher* hay *event* nào không. Tương tự lớp **Product**, lớp **Order** cũng có phương thức trả về toàn bộ thông tin dưới dạng một đối tượng thuộc lớp **json**. Bên cạnh đó, mối quan hệ giữa lớp **Order** và lớp **Date** là *quan hệ bao hàm phụ thuộc* (composition).



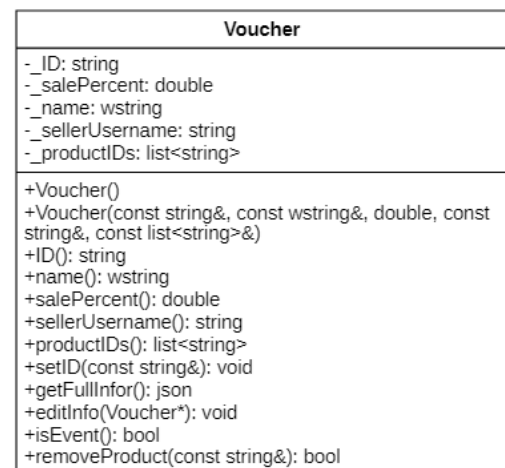
Hình III.2.7. Sơ đồ lớp của lớp Order và lớp Date

III.2.4. Lớp Voucher

Hệ thống có 2 hình thức giảm giá:

- *Event*: Áp dụng cho toàn bộ các sản phẩm hiện có, được tạo bởi các *nhân viên*.
- *Voucher*: Được tạo bởi những *người bán*, áp dụng cho một số sản phẩm họ đã đăng.

Event và *voucher* đều là các đối tượng thuộc lớp **Voucher**. Chúng đều có những thông tin như ID, mức giảm giá (đơn vị phần trăm), tên của *voucher* hay *event*, cũng như tên đăng nhập của người bán và danh sách ID của những sản phẩm được áp dụng. Trong đó, những đối tượng có danh sách ID của



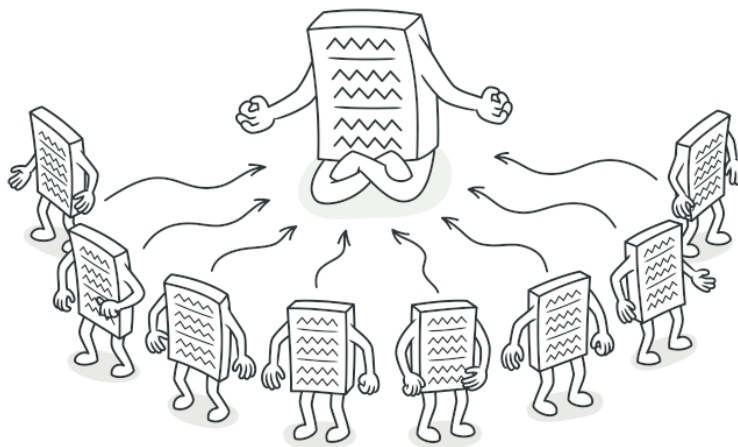
Hình III.2.8. Sơ đồ lớp Voucher

những sản phẩm áp dụng là rỗng thì sẽ được xem là *event*.

Với các *voucher*, ta sẽ có thao tác xóa *ID* của một sản phẩm ra khỏi danh sách *ID* các sản phẩm được áp dụng. Lưu ý rằng, sau khi xóa, nếu danh sách này rỗng thì *voucher* tương ứng cũng sẽ bị xóa ra khỏi hệ thống, công việc này sẽ được hỗ trợ một lớp khác.

III.3. Các lớp “Provider”

Những đối tượng thuộc các lớp **Provider** sẽ cùng đóng vai trò như là các “cơ sở dữ liệu” cho hệ thống, cùng với sự cung cấp một số thao tác liên quan đến các thành phần như *tài khoản*, *sản phẩm*, *đơn hàng* hay là *voucher*, *sự kiện*, đặc biệt là những thao tác phức tạp, cần sự phối hợp giữa nhiều lớp để có thể thực hiện. Để tạo được tính nhất quán cho việc truy xuất, thêm, xóa, sửa các “cơ sở dữ liệu” của từng thành phần, ta hướng đến cách cài đặt sao cho chỉ tồn tại *nhất 1 đối tượng* cho mỗi lớp **Provider** và đối tượng đó có thể được gọi đến một cách *toàn cục*. Từ đó, mẫu thiết kế **Singleton** được áp dụng.



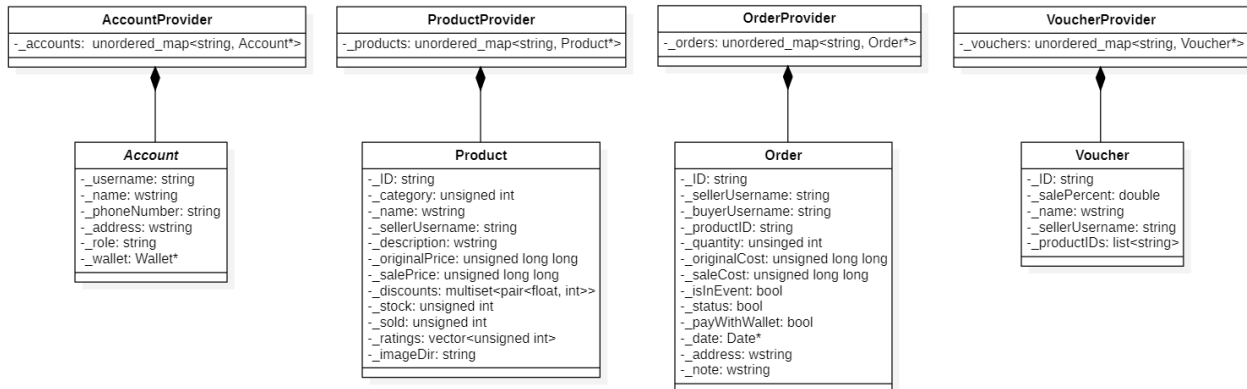
Hình III.3.1. Minh họa mẫu thiết kế Singleton

Thiết kế của hệ thống có tổng cộng 5 lớp **Provider**, bao gồm các lớp **LoginProvider**, **AccountProvider**, **ProductProvider**, **OrderProvider** và **VoucherProvider**. Trong đó, hai lớp **LoginProvider** và **AccountProvider** liên quan nhiều đến lớp **Account**, 3 lớp còn lại lần lượt liên quan nhiều đến lớp **Product**, **Order** và **Voucher**.

Với vai trò như là các “cơ sở dữ liệu”, mỗi lớp **Provider** đều có 2 phương thức là *readData()* và *writeData()*, với những thao tác là đọc dữ liệu lưu trong file JSON tương ứng và ghi lại dữ liệu từ chương trình vào các file JSON đó. Hai phương thức này đều sẽ được gọi lần lượt trong phương thức khởi tạo và phương thức hủy của mỗi lớp **Provider**. Nhằm mục đích lưu trữ dữ liệu đọc được sao cho có thể tối ưu hóa tốc độ truy xuất, thêm, xóa, sửa dữ liệu, cấu trúc dữ liệu **bảng băm** được sử dụng rất nhiều trong việc cài đặt thuộc tính của các lớp **Provider** (thư viện *unordered_map* của **STL C++**), với các cặp *key – value* lần lượt

là định danh của các đối tượng (tên đăng nhập của tài khoản hay ID của sản phẩm, đơn hàng, voucher, sự kiện) và con trỏ đến đối tượng thuộc lớp thành phần tương ứng.

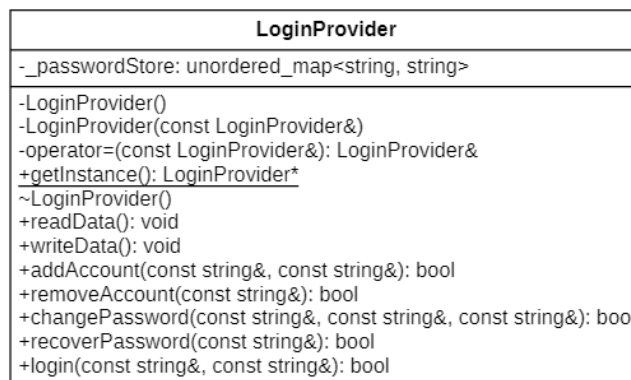
Các đối tượng thuộc lớp thành phần sẽ được tạo ra dựa trên dữ liệu đọc được trong phương thức khởi tạo của lớp **Provider** tương ứng và chúng được giải phóng trong phương thức hủy của lớp đó. Do vậy, ta sẽ có các *quan hệ bao hàm phụ thuộc (composition)* giữa lớp **AccountProvider** và **Account**, **ProductProvider** và **Product**, **OrderProvider** và **Order**, **VoucherProvider** và **Voucher**.



Hình III.3.2. Mối quan hệ composition giữa các lớp Provider và lớp thành phần tương ứng

Tiếp đến, ta sẽ đi vào thiết kế sơ lược của từng lớp **Provider**.

III.3.1. Lớp LoginProvider



Hình III.3.3. Sơ đồ lớp LoginProvider

Lớp **LoginProvider** cung cấp các dữ liệu và những phương thức cần thiết cho các thao tác liên quan đến việc đăng ký, đăng nhập của hệ thống, bao gồm:

- Đọc dữ liệu từ file *passwords.json* và lưu trữ thông tin đăng nhập (tên đăng nhập và mật khẩu) của các tài khoản trong hệ thống dưới dạng **bảng băm** với các cặp key - value là tên đăng nhập - mật khẩu (đã qua mã hóa SHA256).

- Cập nhật lại dữ liệu trong file *passwords.json* khi kết thúc chương trình.
- Thêm, xóa thông tin đăng nhập của một tài khoản nào đó.
- Hỗ trợ thay đổi mật khẩu, tìm lại mật khẩu.
- Kiểm tra thông tin đăng nhập có hợp lệ hay không.

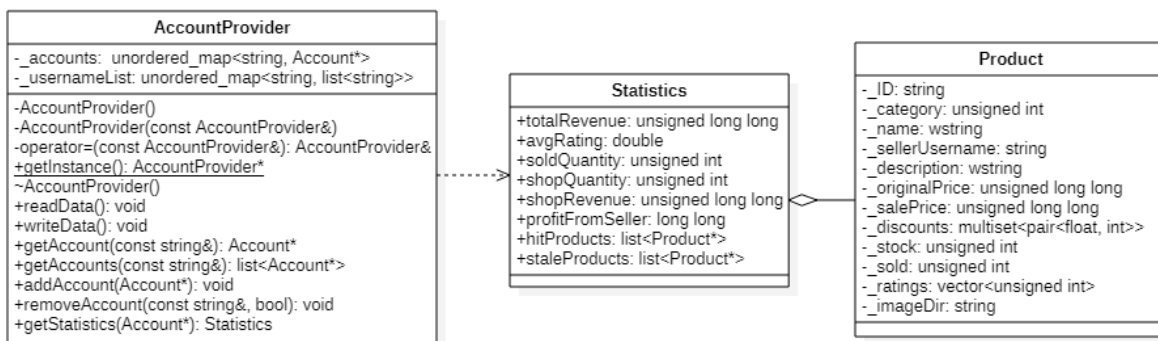
III.3.2. Lớp AccountProvider

Những dữ liệu cần thiết và một số thao tác liên quan đến những tài khoản trong hệ thống sẽ được lớp **AccountProvider** cung cấp. Cụ thể như sau:

- Đọc thông tin của các tài khoản được lưu trong file *accounts.json*, từ đó tạo ra các đối tượng là những tài khoản với những thông tin chung và thông tin đặc trưng tương ứng. Ngoài ra, danh sách những *tên đăng nhập* của các tài khoản cũng được lưu theo từng nhóm, phân theo *vai trò* của các tài khoản.
- Cập nhật lại dữ liệu trong file *accounts.json* khi kết thúc chương trình.
- Truy xuất đến một tài khoản (hay là đối tượng kiểu **Account**) dựa theo *tên đăng nhập* của tài khoản đó.
- Trả về một mảng các con trỏ đến các đối tượng kiểu **Account** có cùng một *vai trò* nào đó.
- Thêm, xóa thông tin của tài khoản nhân viên hoặc người bán khỏi hệ thống.
- Thống kê dữ liệu cho những tài khoản của *người mua, người bán* hoặc *quản lý*.

Trong đó, thao tác thống kê dữ liệu cần có sự tương tác giữa lớp **AccountProvider** và một số lớp **Provider** khác, ta sẽ đề cập cụ thể trong phần **III.5**. Kết quả trả về của thao tác này là một đối tượng thuộc lớp **Statistics**, từ đó *mối quan hệ phụ thuộc (dependency)* giữa lớp **AccountProvider** và lớp **Statistics** được tạo ra.

Với mục đích chỉ làm cầu nối trung gian, đưa kết quả thống kê đến một lớp khác để hiển thị thông tin, lớp **Statistics** được cài đặt theo dạng toàn bộ thuộc tính đều có tầm vực là *public*. Hơn nữa, do nhu cầu lưu trữ danh sách các con trỏ đến một số đối tượng thuộc lớp **Product**, lớp **Statistics** sẽ có *quan hệ bao hàm độc lập (aggregation)* với lớp **Product**.



Hình III.3.4. Sơ đồ lớp của lớp AccountProvider, Statistics và mối quan hệ giữa lớp Statistics với lớp Product

III.3.3. Lớp OrderProvider

Lớp OrderProvider cung cấp các thuộc tính và phương thức hỗ trợ tốt việc quản lý đơn hàng trong hệ thống. Cụ thể như sau:

- Đọc thông tin của các đơn hàng được lưu trong file *orders.json*, từ đó tạo ra các đối tượng thuộc lớp **Order** và lưu trữ dữ liệu này dưới dạng **bảng băm** với các cặp *key – value* là *ID đơn hàng* và *con trỏ kiểu Order*. Bên cạnh đó, số lượng các đơn hàng đã được tạo ra trong hệ thống cũng được lưu trữ.
- Cập nhật lại dữ liệu trong file *orders.json* khi kết thúc chương trình.
- Truy xuất đến một đơn hàng (hay là đối tượng thuộc lớp **Order**) dựa theo ID của đơn hàng đó.
- Thêm, xóa một đơn hàng nào đó trong dữ liệu. Trong đó, với thao tác thêm đơn hàng, ID của đơn hàng này sẽ được cập nhật dựa theo số đơn hàng đã được tạo ra trong hệ thống.
- Xử lý dữ liệu khi một đơn hàng được duyệt hay từ chối bởi *người bán*, bị hủy bởi *người mua*.
- Xóa những đơn hàng liên quan đến một sản phẩm nào đó. Thao tác này được thực hiện khi một sản phẩm bị *người bán* xóa khỏi cửa hàng.
- Tạo ra các đơn hàng cho danh sách những sản phẩm cùng số lượng cho trước.
- Xử lý dữ liệu khi một đơn hàng nhận được đánh giá từ *người mua*.

Trong đó, các thao tác thêm, xóa, đơn hàng, việc xử lý dữ liệu khi đơn hàng được duyệt, bị hủy hay được đánh giá có sự phối hợp thực hiện giữa lớp **OrderProvider** với một số lớp **Provider** và lớp thành phần khác. Ta sẽ đề cập đến vấn đề này trong phần **III.5. Mối quan hệ giữa các lớp**.

OrderProvider
-_orders: unordered_map<string, Order*> -_counter: unsigned int
-OrderProvider() -OrderProvider(const OrderProvider&) -operator=(const OrderProvider&): OrderProvider& <u>+getInstance(): OrderProvider*</u> ~OrderProvider() +readData(): void +writeData(): void -nextID(): string +getOrder(const string&): Order* +addOrder(Order*): void +acceptOrder(const string&): bool +removeOrder(const string&): bool +removeOrdersByProduct(const string&): void +checkoutHandling(const string&, const string&, unsigned int, bool, const wstring&, const wstring&): pair<int, string> +rateOrder(string, unsigned int): void

Hình III.3.5. Sơ đồ lớp OrderProvider

III.3.4. Lớp ProductProvider

Lớp **Product Provider** chứa các phương thức và thuộc tính liên quan đến việc quản lý và thao tác với các sản phẩm. Trong đó bao gồm:

- Đọc dữ liệu về thông tin của các sản phẩm từ file *products.json* và nội dung chính của các sản phẩm trong file *MetaData.txt* và lưu vào 2 **bảng băm**: Các cặp *key - value* trong mỗi bảng lần lượt là *ID của sản phẩm* và *con trỏ kiểu **Product***, *ID* và *nội dung chính* của sản phẩm đó. Đồng thời, những thông tin như số lượng sản phẩm đã được đăng, danh sách sản phẩm phân theo từng danh mục cũng được lưu trữ.
- Ghi dữ liệu vào file *products.json* và *MetaData.txt* khi kết thúc chương trình.
- Truy xuất đến một sản phẩm (hay là đối tượng kiểu **Product**) dựa theo *ID* của sản phẩm đó
- Trả về danh sách các con trỏ đến những đối tượng thuộc kiểu **Product**, dựa theo danh mục của các sản phẩm tương ứng.
- Thêm, xóa sản phẩm khỏi dữ liệu và chỉnh sửa thông tin sản phẩm. Trong thao tác thêm sản phẩm, *ID* của sản phẩm đó sẽ được cập nhật dựa trên số lượng sản phẩm đã đăng trên cửa hàng.
- Tìm kiếm sản phẩm dựa theo chuỗi thông tin mà người dùng cung cấp. Trong đó, thuật toán được áp dụng cho thao tác tìm kiếm sản phẩm là đếm lần xuất hiện của mỗi từ trong chuỗi tìm kiếm với nội dung chính của các sản phẩm (được đọc từ file *MetaData.txt* và lưu vào một **bảng băm**), từ đó đưa ra danh sách sản phẩm liên quan theo thứ tự độ trùng khớp giảm dần.

ProductProvider
-_categoryNames: vector<string> -_products: unordered_map<string, Product*> -_searchingData: unordered_map<string, string> -_categories: unordered_map<string, list<string>> -_counter: unsigned int
-ProductProvider() -ProductProvider(const ProductProvider&) -operator=(const ProductProvider&): ProductProvider& <u>+getInstance(): ProductProvider*</u> ~ProductProvider() +readData(): void +writeData(): void -nextID(): string +getCategoryName(unsigned int): string +addProduct(Product*): void +removeProduct(const string&): bool +changeCategory(const string&, unsigned int, unsigned int): void +updateSearchingData(const string&): void +getProduct(const string&): Product* +getProductsByCategory(const string&): list<string> +getProductsByInput(wstring): list<Product*> +getAllProducts(): list<Product*> +editProduct(const string&, unsigned int, const wstring&, const wstring&, unsigned long long, unsigned int, const string&): void

Hình III.3.6. Sơ đồ lớp ProductProvider

Các thao tác thêm, xóa, sửa sản phẩm cũng đồng thời cập nhật lại dữ liệu dùng cho thao tác tìm kiếm sản phẩm. Ngoài ra, chúng còn có tương tác với một số lớp thành phần là lớp **Buyer** và lớp **Seller**. Ta sẽ đề cập đến trong phần **III.5. Mối quan hệ giữa các lớp**.

III.3.5. Lớp VoucherProvider

Để cho thuận tiện, ta sẽ gọi chung *voucher* và *event* là “mã giảm giá”. Việc quản lí và sử dụng các mã giảm giá được hỗ trợ thực hiện thông qua lớp **VoucherProvider**:

- Đọc dữ liệu từ file *vouchers.json* và lưu trữ thông tin dưới dạng **bảng băm** với cặp *key – value* là *ID* và *thông tin* của mã giảm giá tương ứng. Đồng thời, ID của các *event* cũng sẽ được lưu trữ vào một **danh sách liên kết**.
- Ghi dữ liệu trở lại file *vouchers.json* khi kết thúc chương trình.
- Thêm, xóa sản phẩm khỏi dữ liệu và chỉnh sửa thông tin mã giảm giá. Cũng tương tự như các lớp **Provider** khác, trong thao tác thêm, ID của chúng cũng sẽ được cập nhật dựa trên số lượng mã giảm giá đã được tạo ra trong hệ thống.
- Xóa *ID* của một sản phẩm ra khỏi danh sách các sản phẩm được áp dụng của các *voucher* trong hệ thống. Trong thao tác này, nếu sau khi xóa, một *voucher* nào đó có danh sách sản phẩm áp dụng là rỗng thì nó cũng sẽ bị xóa khỏi dữ liệu.

VoucherProvider
-_vouchers: unordered_map<string, Voucher*> -_events: list<string> -_counter: unsigned int
-VoucherProvider() -VoucherProvider(const VoucherProvider&) -operator(const VoucherProvider&): VoucherProvider& <u>+getInstance(): VoucherProvider*</u> ~VoucherProvider() +readData(): void +writeData(): void -nextID(): string +lastestEvent(): Voucher* +events(): list<string> +getVoucher(const string&): Voucher* +updatePricesWithVoucher(Voucher*, bool): void +addVoucher(Voucher*): void +removeVoucher(const string&): bool +editVoucher(const string&, const wstring&, double, list<string>): void +removeProductFromVouchers(const string&): void

Hình III.3.7. Sơ đồ lớp VoucherProvider

Thao tác thêm, xóa hay chỉnh sửa thông tin mã giảm giá sẽ có liên quan đến một số lớp thành phần là lớp **Product** và lớp **Seller**. Ta sẽ đề cập đến vấn đề đó trong phần **III.5. Mối quan hệ giữa các lớp**.

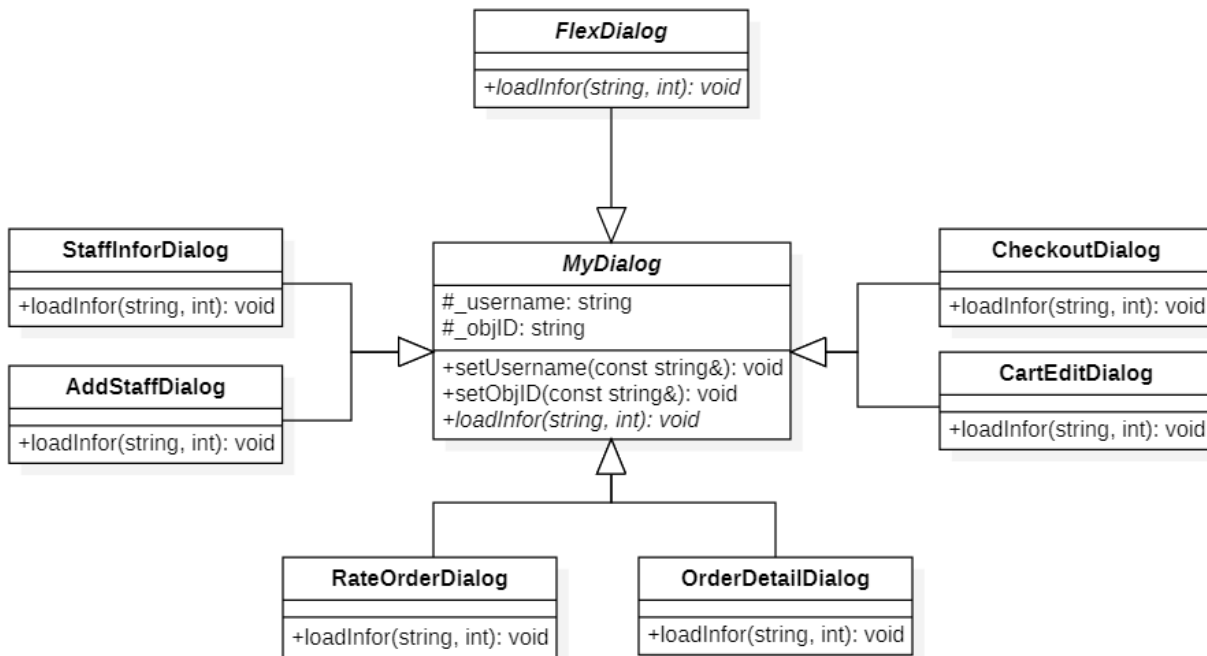
III.4. Các lớp giao diện

Trong phần này, ta sẽ đề cập về những lớp liên quan đến việc hiển thị giao diện cho ứng dụng. Những thuộc tính và phương thức liên quan đến đặc trưng xử lý giao diện của công cụ hỗ trợ mà nhóm sử dụng (**Qt Framework**) sẽ được bỏ qua, ta chỉ xét đến những thuộc

tính và phương thức mang tính cốt lõi. Việc truy xuất đến các đối tượng để lấy thông tin và hiển thị của các lớp giao diện được thực hiện thông qua một số phương thức của các lớp **Provider**. Ngoài ra, những sự tương tác giữa các lớp giao diện với một số lớp **Provider** trong việc xử lý dữ liệu khi một số thao tác cụ thể được thực hiện sẽ được ta đề cập trong phần **III.5. Mối quan hệ giữa các lớp**.

III.4.1. Lớp MyDialog và các lớp kế thừa

Lớp **MyDialog** là lớp cha của các lớp giao diện phụ của ứng dụng, trong đó những lớp con trực tiếp của nó bao gồm các lớp **FlexDialog**, **RateOrderDialog**, **OrderDetailDialog**, **CheckoutDialog**, **CartEditDialog**, **StaffInforDialog** và **AddStaffDialog**. Đây là *lớp trừu tượng* với hàm thuần ảo *loadInfor*, được các lớp kế thừa sử dụng để hiển thị các thông tin tương ứng. Ngoài ra, nó lưu trữ một số thông tin mà hầu hết các lớp giao diện khác đều có sử dụng đến, đó là *tên đăng nhập* của tài khoản đang thực hiện thao tác và *ID* của một đối tượng (*sản phẩm, đơn hàng hay voucher, event*) liên quan đến thao tác đó.



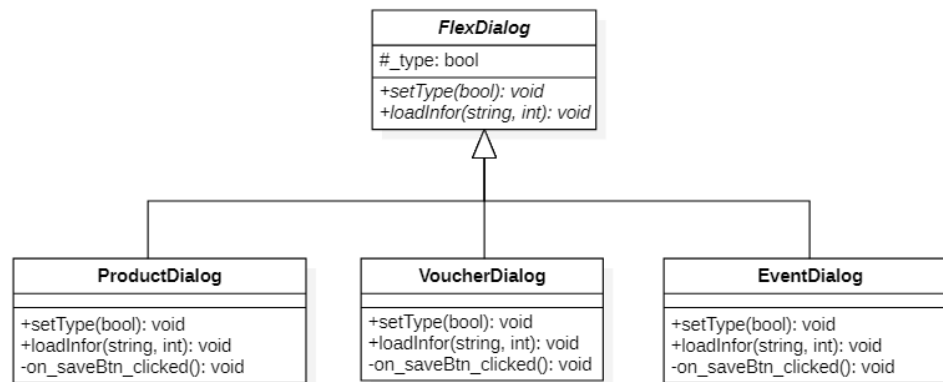
Hình III.4.1. Lớp MyDialog và các lớp kế thừa

Tiếp theo, ta sẽ cùng đi vào chi tiết các lớp giao diện kế thừa trực tiếp từ lớp **MyDialog**.

III.4.1.1. Lớp FlexDialog và các lớp ProductDialog, VoucherDialog, EventDialog

Lớp **FlexDialog** sẽ tùy vào loại thao tác (thuộc cùng một nhóm thao tác) mà hiển thị giao diện tương ứng với loại đó. Việc xác định loại thao tác được thực hiện thông qua thuộc tính *_type* của lớp này, cùng với *phương thức ảo setType*. Phương thức thuần ảo *loadInfor* từ lớp **MyDialog** vẫn giữ nguyên tính chất của nó trong lớp **FlexDialog**. Do vậy, đây cũng là *lớp trừu tượng*.

Điểm chung của các lớp **ProductDialog**, **VoucherDialog** và **EventDialog** là tùy vào thao tác hiện tại đang là thêm mới hay chỉnh sửa thông tin để hiển thị giao diện phù hợp, từ đó, chúng kế thừa lớp **FlexDialog**.



Hình III.4.2. Sơ đồ lớp *FlexDialog*, *ProductDialog*, *VoucherDialog* và *EventDialog*

Lớp **ProductDialog** hỗ trợ phần giao diện cho thao tác thêm và chỉnh sửa thông tin sản phẩm. Trong trường hợp chỉnh sửa sản phẩm, phương thức *loadInfor* của lớp sẽ có vai trò là hiển thị thông tin ban đầu của sản phẩm, bằng cách truy xuất đến đối tượng sản phẩm đó thông qua *ID sản phẩm* để lấy thông tin. Bên cạnh đó, *người bán* đang thực hiện thao tác này và *ID của sản phẩm* được chỉnh sửa sẽ được xác định từ thông tin của lớp **MyDialog**.

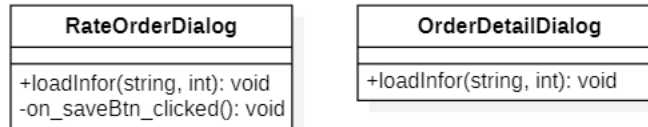
Hai lớp **VoucherDialog** và **EventDialog** hỗ trợ giao diện cho thao tác thêm và chỉnh sửa thông tin của các *voucher* và *event*. Phương thức *loadInfor* trong hai lớp này cũng đóng vai trò hiển thị thông tin ban đầu của *voucher* (hay *event*) cho thao tác chỉnh sửa thông tin. Hơn nữa, *người bán* hay *nhân viên* đang thực hiện thao tác hiện tại cũng như *ID của voucher* hay *event* được chỉnh sửa sẽ được xác định từ thông tin của lớp **MyDialog**.

III.4.1.2 Lớp **RateOrderDialog** và **OrderDetailDialog**

Hai lớp **RateOrderDialog** và **OrderDetailDialog** sẽ hỗ trợ giao diện cho một số thao tác liên quan đến những đơn hàng, bao gồm xem thông tin chi tiết của đơn hàng và đánh giá đơn hàng.

Lớp **RateOrderDialog** sẽ nhận điểm số đánh giá của người mua thông qua một hộp thoại và chuyển giá trị đó đến một thao tác khác, đồng thời lưu trữ *ID* của đơn hàng đang được đánh giá để hiển thị nó cho người dùng.

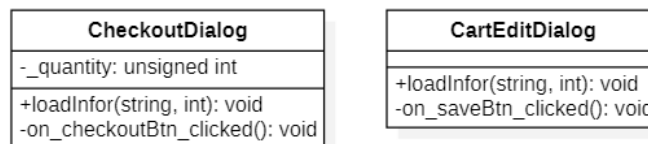
Lớp **OrderDetailDialog** sẽ truy xuất đến đối tượng thuộc kiểu **Order** thông qua *ID* của đơn hàng đó, lấy thông tin và hiển thị ra giao diện với phương thức *loadInfor*. Lưu ý rằng, *ID* của các đơn hàng mà ta đề cập được lưu trữ trong thông tin của lớp **MyDialog**.



Hình III.4.3. Sơ đồ lớp *RateOrderDialog* và *OrderDetailDialog*

III.4.1.4. Lớp *CheckoutDialog* và *CartEditDialog*

Hai lớp **CheckoutDialog** và **CartEditDialog** đều hỗ trợ giao diện cho các thao tác của đối tượng là *người mua*. Lớp **CheckoutDialog** hiển thị giao diện cho quá trình thanh toán, bao gồm hỗ trợ người mua cung cấp thông tin như địa chỉ, ghi chú giao hàng, kiểm tra lại những sản phẩm sẽ mua và lựa chọn hình thức thanh toán. Mặt khác, lớp **CartEditDialog** hỗ trợ cho việc thay đổi số lượng của một sản phẩm trong giỏ hàng.

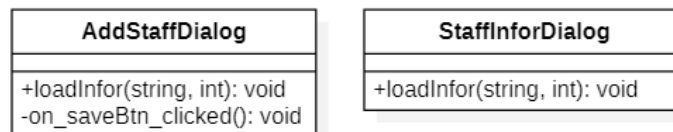


Hình III.4.4. Sơ đồ lớp *CheckoutDialog* và *CartEditDialog*

Việc thanh toán sẽ được thực hiện theo một trong hai tình huống là mua ngay một sản phẩm nào đó hoặc thanh toán các sản phẩm trong giỏ hàng. Do đó, lớp **CheckoutProduct** có sử dụng thêm một thuộc tính để xác định điều đó. Ngoài ra, lớp **CartEditDialog** có thể xác định được *ID* của sản phẩm mà người mua muốn thay đổi số lượng trong giỏ hàng (từ thông tin của lớp **MyDialog**), từ đó gửi thông tin như số lượng mới và *ID* của sản phẩm đó cho một thao tác khác.

III.4.1.5. Lớp *StaffInforDialog* và *AddStaffDialog*

Hai lớp **StaffInforDialog** và **AddStaffDialog** hỗ trợ giao diện cho những thao tác liên quan đến *nhân viên* mà *quản lý* của hệ thống có thể thực hiện. Phương thức *loadInfor* của lớp **StaffInforDialog** hiển thị thông tin về một nhân viên bằng cách truy xuất đến đối tượng thuộc lớp **Staff** thông qua *tên đăng nhập* của nhân viên đó. Lớp **AddStaffDialog** hỗ trợ việc thêm một *người bán* nào đó thành *nhân viên*.



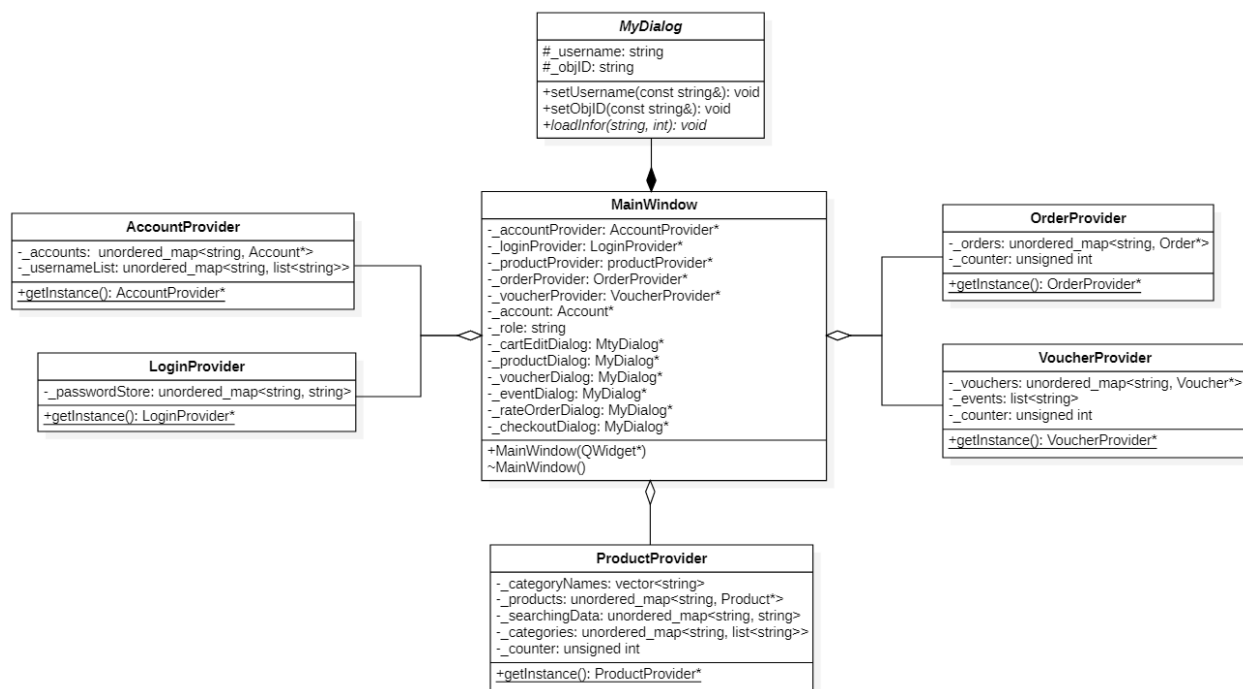
Hình III.4.5. Sơ đồ lớp *AddStaffDialog* và *StaffInforDialog*

III.4.2. Lớp *MainWindow*

Lớp **MainWindow** là lớp giao diện cho cửa sổ chính của ứng dụng. Lớp này sẽ bao hàm tất cả các lớp giao diện khác, thông qua lớp **MyDialog**, cùng với các lớp **Provider**. Trong

đó, mối quan hệ giữa lớp **MainWindow** với lớp **MyDialog** là *quan hệ bao hàm phụ thuộc (composition)* và với các lớp **Provider** là *quan hệ bao hàm độc lập (aggregation)*.

Bên cạnh đó, lớp **MainWindow** còn có những thuộc tính liên quan đến tài khoản đang đăng nhập trong hệ thống, bao gồm *con trỏ* đến đối tượng thuộc lớp **Account** và vai trò của tài khoản đó (để thuận tiện cho các thao tác trong ứng dụng).



Hình III.4.6. Sơ đồ lớp **MainWindow** cùng quan hệ bao hàm của nó với lớp **MyDialog** và các lớp **Provider**

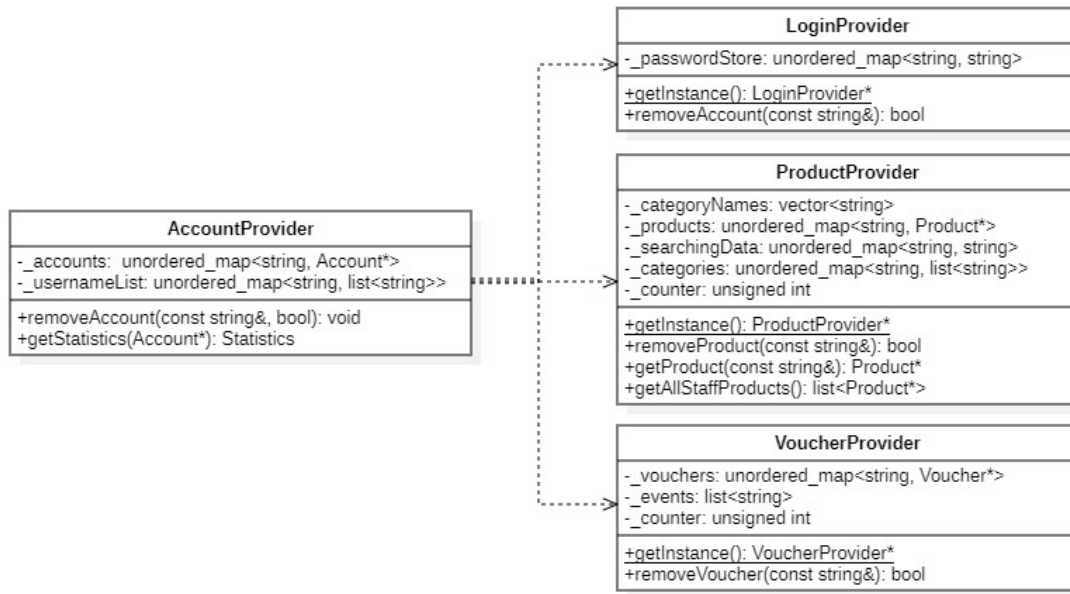
Ngoài ra, lớp **MainWindow** có khá nhiều những thuộc tính và phương thức liên quan đến việc xử lý giao diện như hiển thị thông tin ứng với các sự kiện ứng dụng nhận được từ người dùng như là *click chuột*. Như đã nêu ra từ trước, ta sẽ không đề cập đến những thuộc tính, phương thức đó mà chỉ chú trọng vào các thuộc tính, phương thức cốt lõi.

III.5. Môi quan hệ giữa các lớp

III.5.1. Giữa các lớp **Provider**

III.5.1.1. **AccountProvider** và **LoginProvider**, **ProductProvider**, **VoucherProvider**

Lớp **AccountProvider** cần sự trợ giúp từ các lớp **LoginProvider**, **ProductProvider** và **VoucherProvider** để có thể thực hiện một số phương thức của nó. Từ đó, ta có mối quan hệ phụ thuộc (*dependency*) giữa **AccountProvider** và những lớp đó.



Hình III.5.1. Quan hệ dependency giữa lớp *AccountProvider* với các lớp *LoginProvider*, *ProductProvider* và *VoucherProvider*

Cụ thể về sự phụ thuộc trên như sau:

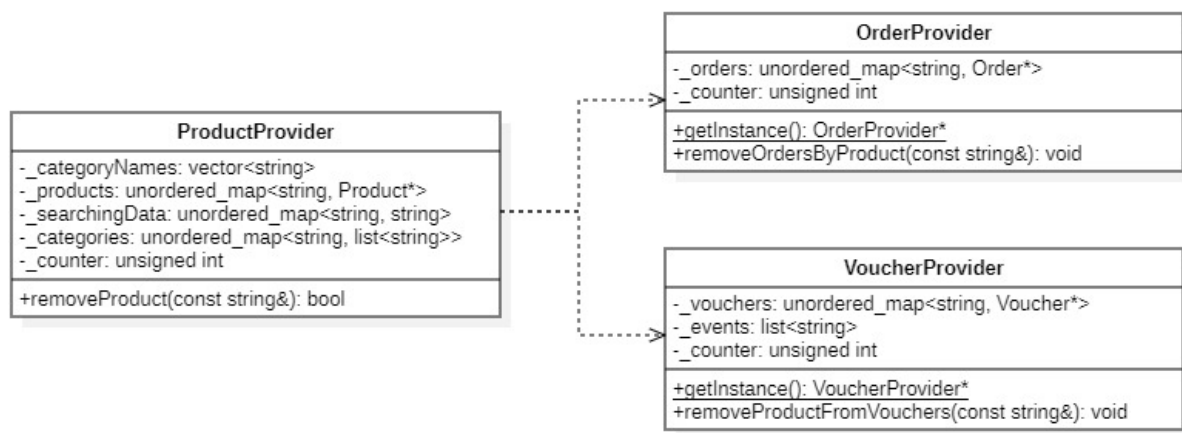
- Lớp **AccountProvider** có hỗ trợ thao tác xóa thông tin của một tài khoản người mua hoặc nhân viên khỏi hệ thống, thông qua phương thức *removeAccount* (phương thức này được gọi khi *quản lý* của hàng thực hiện thêm *nhân viên* hoặc thao tác sa thải *nhân viên* hay hủy hợp đồng với *người bán* (chi tiết về các thao tác đó sẽ được đề cập ở phần III.5.2).
Khi đối tượng cần xóa là *nhân viên* hoặc *người bán*, **AccountProvider** sẽ thực hiện việc xóa toàn bộ *voucher* (hay *event*) và *sản phẩm* có liên quan đến tài khoản đó thông qua các phương thức *removeVoucher* và *removeProduct* (chúng được cài đặt trong các lớp **VoucherProvider**, **ProductProvider**).
Lưu ý rằng, trong thao tác *thêm nhân viên*, ta chỉ xóa các *voucher* đã được tạo từ trước bởi tài khoản *người bán* tương ứng chứ không xóa các *sản phẩm* họ đã đăng. Tiếp đến, thông tin đăng nhập của tài khoản đó cũng bị xóa khỏi hệ thống thông qua phương thức *removeAccount* được cài đặt trong lớp **LoginProvider**.
- Ngoài ra, ở đây ta không thực hiện thao tác xóa các *đơn hàng* liên quan đến tài khoản này. Lý do là điều đó sẽ được thực hiện khi những *sản phẩm* liên quan đến *đơn hàng* đó bị xóa khỏi dữ liệu, ta sẽ đề cập đến thao tác này ở phần kế tiếp (III.5.1.2).
- Lớp **AccountProvider** còn hỗ trợ việc thống kê những thông tin liên quan đến doanh thu của *người bán*, *nhân viên* hoặc là *quản lý*, thông qua phương thức *getStaticstics*, bao gồm tổng doanh thu, mức độ hài lòng của khách hàng, số sản phẩm đã đăng và đã bán được, những sản phẩm bán chạy hoặc bán chậm. Để có

được những thông tin đó, phương thức trên cần sử dụng những phương thức như *getProduct*, *getAllStaffProduct* được cài đặt trong **ProductProvider** để lấy những thông tin tương ứng, từ đó tiến hành quá trình tính toán và thống kê.

III.5.1.2. ProductProvider và OrderProvider, VoucherProvider

Lớp **ProductProvider** phụ thuộc vào lớp **OrderProvider**. Cụ thể, khi cần loại bỏ một sản phẩm (do *người bán* hoặc *nhân viên* xóa), ta cần xóa những đơn hàng liên quan đến sản phẩm đó. Ngoài ra, ta cũng sẽ loại sản phẩm này ra khỏi danh sách sản phẩm được giảm giá của một *voucher* nào đó.

Để có thể thực hiện điều đó, lớp **ProductProvider** cần phải gọi đến những phương thức *removeOrdersByProduct* được cài đặt trong lớp **OrderProvider** và phương thức *removeProductFromVouchers* trong lớp **VoucherProvider**.



Hình III.5.2. Quan hệ dependency giữa lớp *ProductProvider* và các lớp *OrderProvider*, *VoucherProvider*

III.5.2. Giữa các lớp Provider và các lớp thành phần

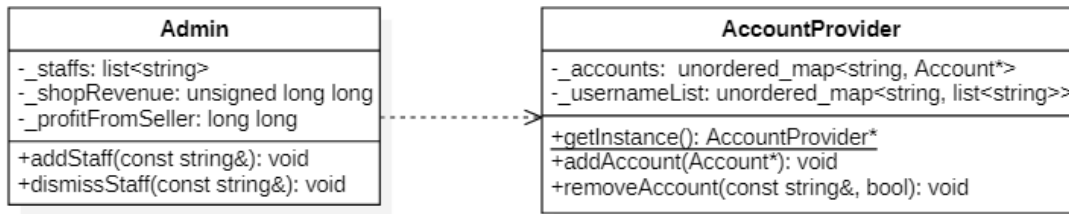
Trong phần này ta chỉ xét các *mối quan hệ phụ thuộc (dependency)* giữa các lớp **Provider** và các lớp thành phần (các mối quan hệ *composition* đã được đề cập ở phần III.3. Các lớp “Provider”).

III.5.2.1. Lớp AccountProvider và lớp Admin

Liên quan đến lớp **Admin**, khi *quản lý* thêm một *người bán* làm *nhân viên* (ứng với phương thức *addStaff*), ta tạo một đối tượng nhân viên mới với cùng thông tin cơ bản giống với người bán, sau đó tiến hành xóa thông tin tài khoản người bán đó và thêm thông tin của tài khoản nhân viên mới này vào dữ liệu, thông qua các phương thức *removeAccount* và *addAccount* trong lớp **AccountProvider**.

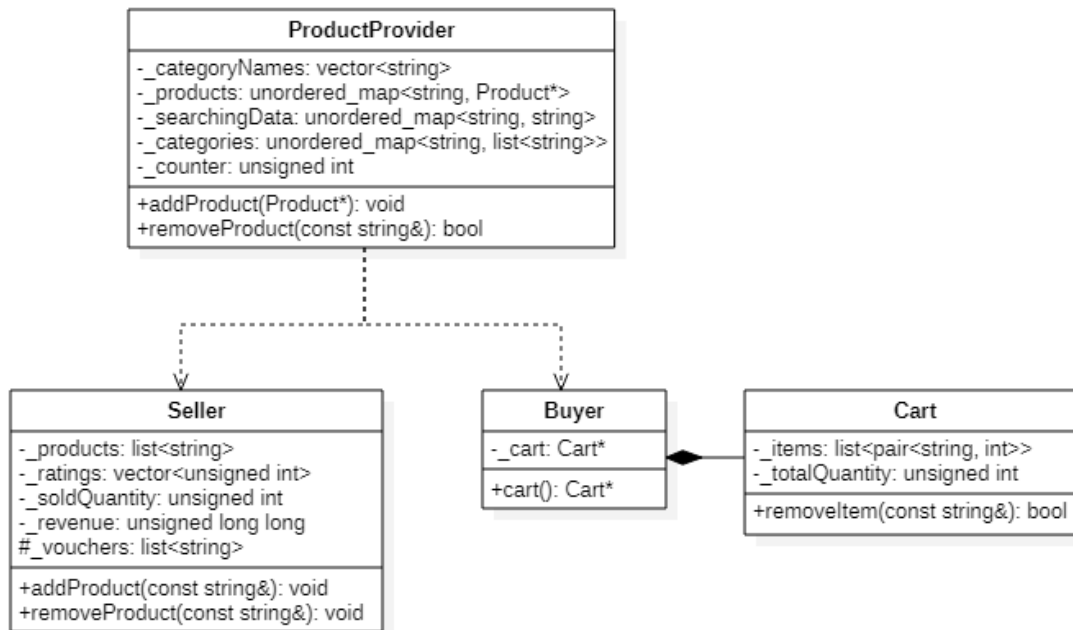
Tương tự, khi *quản lý* sa thải một *nhân viên* (ứng với phương thức *dismissStaff*), tức toàn bộ thông tin về tài khoản của nhân viên đó sẽ bị xóa khỏi hệ thống, chương trình sẽ tiến

hành gọi đến phương thức *removeAccount* trong lớp **AccountProvider** để thực hiện công việc này (lớp **AccountProvider** sẽ kết hợp với lớp **LoginProvider** để xóa cả thông tin đăng nhập của tài khoản đó). Do đó lớp **Admin** phụ thuộc vào lớp **AccountProvider**.



Hình III.5.3. Quan hệ dependency giữa lớp *AccountProvider* và lớp *Admin*

III.5.2.2. Lớp **ProductProvider** và các lớp **Seller**, **Buyer**



Hình III.5.4. Quan hệ dependency giữa lớp *ProductProvider* và các lớp *Seller*, *Buyer*

Ở phần này chúng ta phân tích về 2 thao tác được lớp **ProductProvider** hỗ trợ, bao gồm *thêm sản phẩm*, *xóa sản phẩm*, ứng với các phương thức *addProduct*, *removeProduct* của lớp này. Cụ thể như sau:

- **Thêm sản phẩm:** Khi một sản phẩm mới được thêm vào hệ thống, dữ liệu sử dụng cho thuật toán tìm kiếm sẽ được cập nhật, đồng thời sản phẩm đó sẽ được thêm vào danh sách sản phẩm của hệ thống và của người bán tương ứng. Lúc này phương thức *addProduct* trong lớp **Seller** cũng sẽ được gọi bởi lớp **ProductProvider**. Do đó, lớp **ProductProvider** phụ thuộc vào lớp **Seller**.

- **Xóa sản phẩm:** Khi người bán hay nhân viên thực hiện việc xóa một sản phẩm trên hệ thống, sản phẩm đó sẽ bị xóa đi khỏi 4 danh sách:
 - Các sản phẩm trên hệ thống
 - Những đơn hàng liên quan đến sản phẩm đó
 - Những mặt hàng người bán đã đăng
 - Các sản phẩm đang có trong giỏ hàng của người mua

Trong đó, khi xóa sản phẩm trong danh sách những mặt hàng người bán đã đăng, hàm *removeProduct* thuộc lớp **Seller** sẽ được gọi và hàm này nhận vào *ID* của sản phẩm đó, nhằm xóa hoàn toàn sản phẩm ra khỏi danh sách nói trên. Tương tự, phương thức *cart* thuộc lớp **Buyer** cũng được gọi nhằm truy xuất vào giỏ hàng của người mua, từ đó sản phẩm cũng được xóa hoàn toàn khỏi giỏ hàng. Do đó, lớp **ProductProvider** sẽ phụ thuộc vào lớp **Buyer** và tiếp tục phụ thuộc vào lớp **Seller**.

III.5.2.3. Lớp OrderProvider và các lớp Product, Admin, Member, Seller

Ta điếm qua một số hằng số được sử dụng trong chương trình:

- **STAFF_BONUS = 10%:** với mỗi đơn hàng được nhân viên chấp nhận, nhân viên đó sẽ nhận được 10% giá trị đơn hàng đó, quản lý sẽ nhận 90% còn lại.
- **ORDER_PROFIT = 5%:** với mỗi đơn hàng được người bán (không phải nhân viên) chấp nhận, người bán sẽ nhận được 95% giá trị đơn hàng đó, quản lý sẽ nhận 5% còn lại.

Chúng ta cùng xem xét 2 ví dụ minh họa cho các trường hợp liên quan đến thao tác thanh toán như dưới đây:

- **VD1:** Đơn hàng chứa 1 sản phẩm có giá gốc 100.000 đồng đang áp dụng voucher giảm 10%. Khi đó:
 - Người mua phải trả $100.000 \times (1 - 10\%) = 90.000$ đồng.
 - Người bán (không phải nhân viên) nhận được $90.000 \times (1 - 5\%) = 85.500$ đồng.
 - Quản lý nhận được $90.000 \times 5\% = 4.500$ đồng.
 - Nếu đây là sản phẩm do nhân viên đăng thì nhân viên sẽ nhận được $90.000 \times 10\% = 9.000$ tiền thưởng, quản lý nhận $90.000 \times 90\% = 81.000$ đồng.
- **VD2:** Đơn hàng chứa 1 sản phẩm có giá gốc 100.000 đồng đang áp dụng sự kiện giảm 15% của nhân viên. Khi đó:
 - Người mua phải trả $100.000 \times (1 - 15\%) = 85.000$ đồng.
 - Người bán (không phải nhân viên) nhận được $100.000 \times (1 - 5\%) = 95.000$ đồng.
 - Quản lý phải trả $95.000 - 85.000 = 10.000$ đồng.
 - Nếu đây là sản phẩm do nhân viên đăng thì nhân viên sẽ nhận được $85.000 \times 10\% = 8.500$ đồng tiền thưởng, lúc này quản lý của cửa hàng sẽ nhận được $85.000 \times 90\% = 76.500$ đồng.

Tiếp theo chúng ta đi vào phân tích 5 thao tác được hỗ trợ bởi lớp **OrderProvider**, bao gồm *thanh toán (tức là tạo đơn hàng)*, *thêm đơn hàng mới* vào dữ liệu, *chấp nhận đơn hàng xóa đơn hàng* và *đánh giá đơn hàng*, ứng với các phương thức *checkoutHandling*, *addOrder*, *acceptOrder*, *removeOrder* và *rateOrder*. Cụ thể như sau:

- **Thanh toán (tạo các đơn hàng):** Việc thanh toán được tiến hành trong các tình huống như sau:

- Mua ngay một sản phẩm nào đó
- Thanh toán các sản phẩm có trong giỏ hàng

Để có thể xác định được tình huống cụ thể, phương thức *checkoutHandling* trong lớp **OrderProvider** sử dụng thêm những tham số liên quan đến *ID* của sản phẩm (tham số *productID*) cùng số lượng được mua của sản phẩm đó:

- Khi người mua thực hiện *mua ngay*, đơn hàng chỉ gồm 1 loại sản phẩm duy nhất và *productID* chứa *ID* của sản phẩm đó.
- Ngược lại, khi thanh toán trong giỏ hàng, tất cả các sản phẩm trong giỏ hàng sẽ được thanh toán, khi này *productID* là chuỗi rỗng và ta sẽ gọi đến hàm *cart* trong lớp **Buyer** để lấy ra danh sách sản phẩm trong giỏ hàng và tính tổng giá tiền.

Nếu người mua có sử dụng ví điện tử cho việc thanh toán, số tiền trong ví của họ sẽ bị trừ một lượng tương ứng. Việc kiểm tra số lượng sản phẩm có sẵn trong kho cũng được tiến hành và bắt lỗi nếu số lượng sản phẩm trong kho không đủ. Những đơn hàng tương ứng với các sản phẩm cùng số lượng mỗi sản phẩm sẽ được tạo ra.

Sau đó, ta sẽ xóa những sản phẩm đã thanh toán ra khỏi giỏ hàng trong trường hợp thanh toán những sản phẩm đó, đồng thời tạo ra những đơn hàng mới ứng với từng loại sản phẩm.

- **Thêm đơn hàng mới:** ngay khi tạo *ID* mới cho đơn hàng, đơn hàng mới này sẽ được thêm vào danh sách các đơn hàng đang chờ của *người bán* và *người mua* (ứng với thuộc tính *_pendingOrders*). Do đó phương thức *addOrder* sẽ gọi đến hai phương thức ảo cùng tên trong lớp **Member**, truyền vào *ID* của đơn hàng mới này để thêm vào thuộc tính *_pendingOrders*.
- **Chấp nhận đơn hàng:** Ta sẽ gọi chung *người bán* và *nhân viên* là *người bán*. Khi *người bán* chấp nhận đơn hàng, sẽ có một vài sự thay đổi liên quan đến dữ liệu của sản phẩm trong đơn hàng đó:

- Số lượng sản phẩm đã bán tăng lên
- Số lượng sản phẩm có sẵn trong kho giảm đi

Tuy nhiên, trong thao tác này, ta sẽ chỉ tăng số lượng đã bán (thông qua phương thức *changeSoldQuantity* của lớp **Product**) chứ không giảm số lượng có sẵn trong kho. Bởi vì, nếu số lượng sản phẩm trong kho không đủ để giao cho khách hàng, chương trình sẽ báo lỗi. Lỗi này nên được thông báo cho *người mua* khi họ thanh toán, hơn là khi *người bán* đang chấp nhận đơn hàng.

Sau khi được chấp nhận, đơn hàng đó sẽ được thêm vào danh sách những đơn hàng được chấp nhận của người bán và người mua (ứng với thuộc tính *_acceptedOrders*). Do đó hàm *acceptOrder* sẽ gọi đến hai hàm ảo cùng tên trong lớp **Member**, truyền vào *ID* của đơn hàng để thêm đơn hàng vào thuộc tính *_acceptedOrders*, đồng thời loại bỏ khỏi *_pendingOrders*.

Sau đó, chương trình tiến hành tính doanh thu cho người bán, trong đó bao gồm việc xác định đơn hàng này có phải nhân viên chốt hay không có thể tính lợi nhuận cho quản lý theo đúng công thức (gọi đến phương thức *addShopRevenue* hoặc *addProfitFromSeller* trong lớp **Admin**).

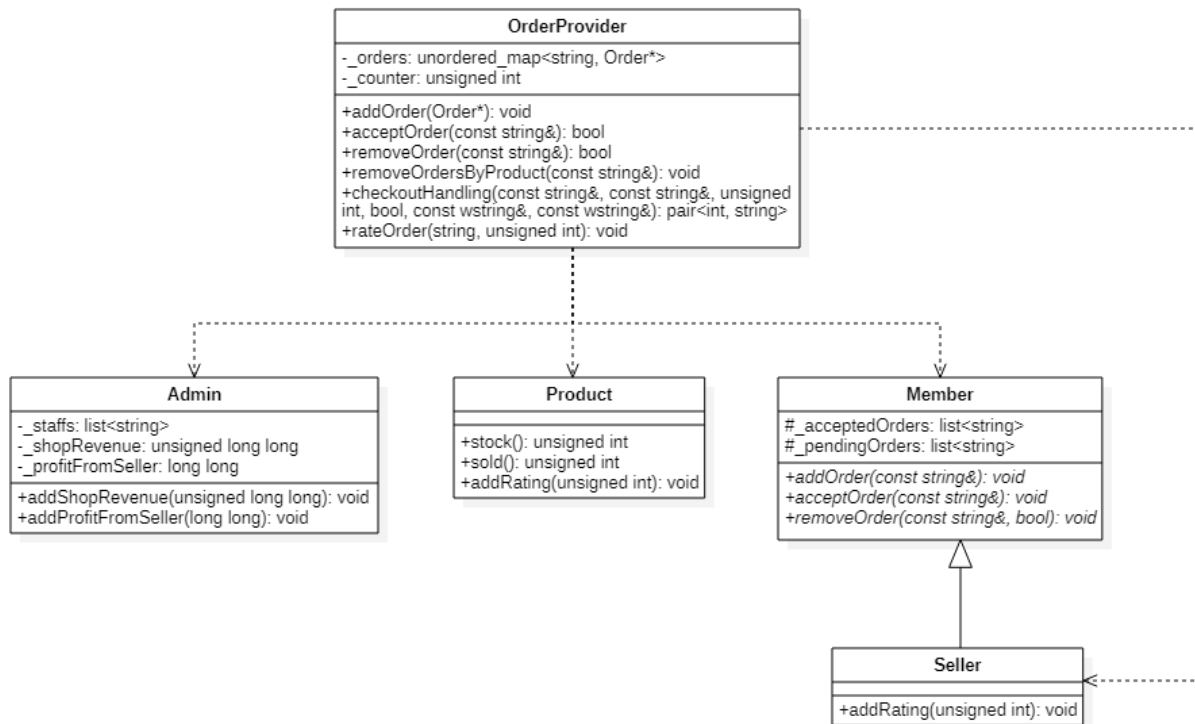
Ta giả thiết rằng sau khi được chấp nhận, đơn hàng luôn được giao thành công đến người mua và người mua có thể đánh giá đơn hàng ngay lập tức.

- **Xóa đơn hàng:** Việc xóa đơn hàng sẽ được tiến hành khi *người bán* (hoặc *nhân viên*) từ chối một đơn hàng, *người mua* hủy đơn hàng hoặc là đơn hàng bị xóa vì một sản phẩm liên quan đã bị xóa.

Tùy vào trạng thái của đơn hàng (đã duyệt hoặc chưa duyệt, ứng với giá trị của thuộc tính *status* của đối tượng tương ứng là *true* hay *false*), ta sẽ có những hướng xử lý khác nhau:

- **Xóa những đơn hàng đã được người bán chấp nhận (*status = true*):** ta chỉ việc loại bỏ đơn hàng này ra khỏi danh sách những đơn hàng có trên hệ thống và những đơn hàng được chấp nhận ở cả người bán và người mua, bằng cách gọi đến phương thức thuần ảo *removeOrder* trong lớp **Member**.
- **Xóa những đơn hàng chưa được người bán chấp nhận (*status = false*):** làm tương tự như trường hợp trên, tuy nhiên ta phải bổ sung thêm một số thao tác khác như gọi đến phương thức *changeStockQuantity* trong lớp **Product** để tăng số lượng sản phẩm trong kho (vì đơn hàng đã bị người mua thu hồi), đồng thời cũng gọi đến phương thức thuần ảo *removeOrder* trong lớp **Member**, trong đó phương thức *removeOrder* trong lớp **Buyer** được ghi đè với thao tác hoàn tiền lại cho *người mua* nếu họ có sử dụng ví điện tử để thanh toán.
- **Đánh giá đơn hàng:** Với thông tin đánh giá nhận được từ người mua, phương thức *rateOrder* trong lớp **OrderProvider** sẽ lần lượt gọi đến phương thức *addRating* trong hai lớp **Product** và **Seller** để đưa đánh giá này vào những thuộc tính lưu các đánh giá đã nhận được của đối tượng *sản phẩm* và *người bán* tương ứng.

Qua các phân tích trên, lớp **OrderProvider** sẽ phụ thuộc vào các lớp **Product**, **Admin**, **Member** và lớp **Seller**.



Hình III.5.5. Quan hệ dependency giữa lớp OrderProvider và lớp Product, Admin, Member

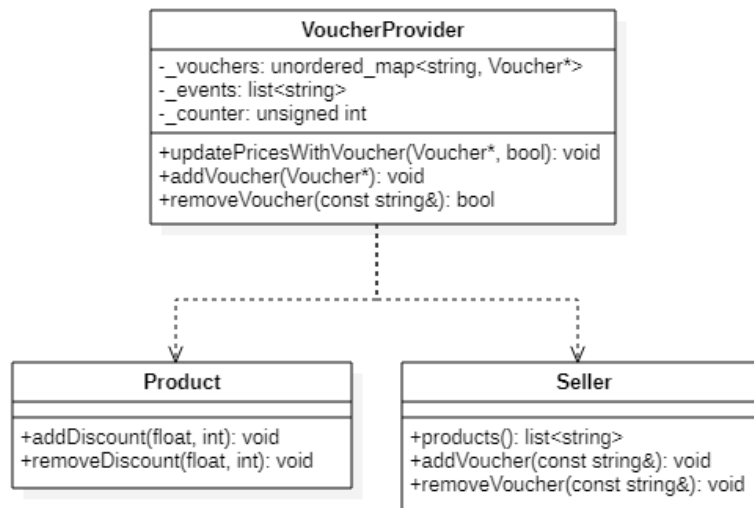
III.5.2.4. Lớp VoucherProvider và các lớp Product, Seller

Trong phần này, nếu không nói gì thêm, ta hiểu *voucher* bao gồm các tất cả các *voucher* do người bán và sự kiện do nhân viên tạo ra. Chúng ta đi vào phân tích 3 thao tác được hỗ trợ bởi lớp **VoucherProvider** là *tạo voucher*, *chỉnh sửa voucher* và *xóa voucher*, ứng với các phương thức *addVoucher*, *editVoucher* và *removeVoucher* được cài đặt trong lớp **VoucherProvider**. Đồng thời, ta cũng sẽ đề cập đến một thao tác quan trọng trước là cập nhật giá bán của những sản phẩm liên quan đến một *voucher* nào đó (ứng với phương thức *updatePricesWithVoucher* trong lớp này). Cụ thể như sau:

- **Cập nhật giá mới cho các sản phẩm áp dụng voucher:** Thao tác này sẽ được thực hiện khi một *voucher* được tạo ra hay bị xóa đi. Để biết được việc cập nhật diễn ra theo hướng thêm một mức giảm giá mới hay bỏ đi mức giảm nào đó, ta sử dụng tham số đầu vào *bool isCreated* để xác định, từ đó có hướng điều chỉnh giá sản phẩm cho phù hợp với các phương thức *addDiscount* và *removeDiscount* trong lớp **Product**. Nếu *voucher* này thực sự là một *event*, chương trình gọi các hàm tương ứng để cập nhật giá mới cho *tất cả các sản phẩm* trong hệ thống. Ngược lại, ta chỉ cần cập nhật giá mới cho một số sản phẩm. Do đó **VoucherProvider** phụ thuộc vào lớp **Product**.
- **Tạo một voucher mới:** Ta sẽ tạo ID cho *voucher* này, sau đó thêm nó vào danh sách các *voucher* hiện có trên hệ thống (thuộc tính *_vouchers* của lớp **VoucherProvider**).

Hiển nhiên ta cần thao tác kiểm tra *voucher* này có phải là một *event* hay không để đồng thời thêm vào danh sách các sự kiện đang diễn ra (thuộc tính *_events*). Tiếp theo, chương trình sẽ tiến hành cập nhật giá các sản phẩm thông qua phương thức *updatePricesWithVoucher*. Sau đó, *ID* của *voucher* này sẽ được thêm vào danh sách các *voucher* của người bán (hay nhân viên) tương ứng, thông qua phương thức *addVoucher* của lớp **Seller**. Do đó, lớp **VoucherProvider** phụ thuộc vào lớp **Seller**.

- **Chỉnh sửa voucher:** Trong thao tác chỉnh sửa thông tin một *voucher*, ta sẽ tiến hành xóa *voucher* cũ và tạo lại một *voucher* mới cùng các thông tin tương ứng. Khi đó, việc cập nhật lại giá bán của các sản phẩm liên quan cũng được diễn ra.
- **Xóa một voucher:** Tương tự như khi tạo *voucher* mới, sau khi xóa *voucher* khỏi danh sách *voucher* trên hệ thống (và danh sách *event* – nếu đây thực sự là một *event*), chương trình sẽ tiến hành cập nhật lại giá bán các sản phẩm. Cuối cùng chương trình gọi đến hàm *removeVoucher* trong lớp **Seller** để xóa *voucher* này khỏi danh sách *voucher* của người bán (hoặc nhân viên).



Hình III.5.6. Quan hệ dependency giữa lớp *VoucherProvider* và các lớp *Product*, *Seller*

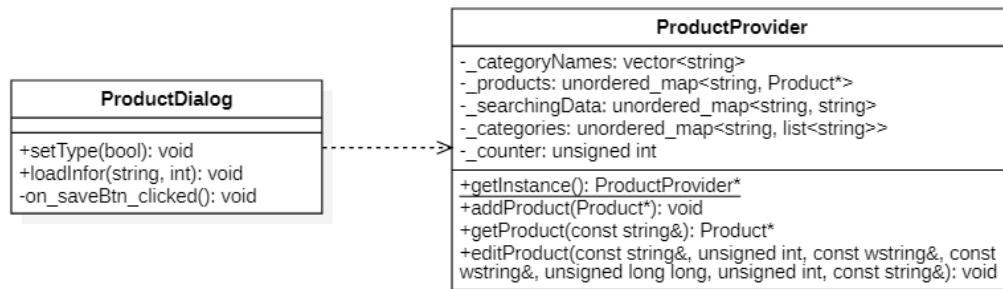
III.5.3. Giữa các lớp **Provider** và các lớp giao diện

Trong phần này, chúng ta sẽ đề cập đến mối quan hệ giữa các lớp **Provider** và các lớp giao diện. Việc hoàn thành các phương thức lớp giao diện cần sự hỗ trợ từ **Provider** đã tạo nên mối quan hệ phụ thuộc (*dependency*) giữa chúng. Chúng ta hãy đi tiếp những phần dưới đây để hiểu rõ sự phụ thuộc này.

III.5.3.1. **ProductProvider** và **ProductDialog**

Lớp **ProductDialog** hỗ trợ giao diện cho hai thao tác chính là thêm và chỉnh sửa thông tin sản phẩm:

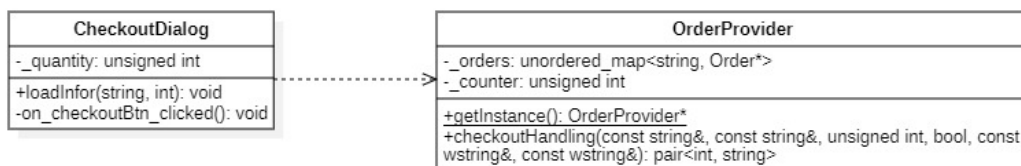
- Khi người bán hoặc nhân viên có nhu cầu chỉnh sửa thông tin sản phẩm, lớp **ProductDialog** sẽ thực hiện phương thức *loadInfor* với tham số đầu vào là *ID* của sản phẩm, đối tượng thuộc lớp **Product** tương ứng được xác định thông qua lớp **ProductProvider**, toàn bộ thông tin hiện tại của sản phẩm tương ứng được truy xuất và hiển thị lên giao diện.
- Để hoàn thành quá trình thêm hoặc lưu thông tin sản phẩm sau khi chỉnh sửa, lớp **ProductDialog** sẽ gọi đến các phương thức của **ProductProvider** tương ứng như *editProduct* với thao tác chỉnh sửa hay *addProduct* với thao tác thêm một sản phẩm mới vào gian hàng.



Hình III.5.7. Mối quan hệ giữa *ProductProvider* và *ProductDialog*

III.5.3.2. OrderProvider và CheckoutDialog

Sau khi người mua điền đầy đủ các thông tin về đơn hàng tại cửa sổ thanh toán, lớp **CheckoutDialog** sẽ thực hiện việc xử lý, tạo các đơn hàng thông qua phương thức *checkoutHandling* được cài đặt trong lớp **OrderProvider**.

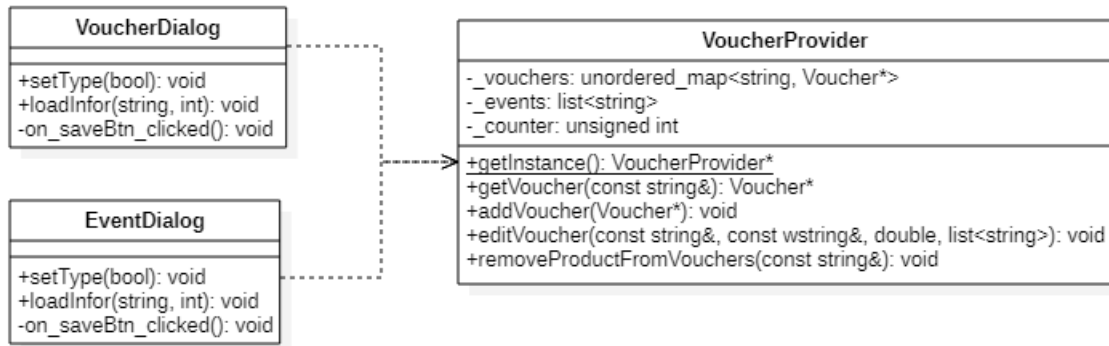


Hình III.5.8. Mối quan hệ giữa *OrderProvider* và *CheckoutDialog*

III.5.3.3. VoucherProvider và VoucherDialog, EventDialog

Tương tự như lớp **ProductDialog**, hai lớp **VoucherDialog** và **EventDialog** hỗ trợ hai thao tác chính là thêm và chỉnh sửa thông tin của *voucher* (*event*).

- Khi ở giao diện chỉnh sửa, phương thức *loadInfor* được cài đặt trong 2 lớp giao diện này sẽ lấy thông tin của *voucher* (hay *event*) để hiển thị, việc lấy thông tin đó sẽ được thực hiện thông qua phương thức *getVoucher* trong lớp **VoucherProvider**
- Để hoàn thành việc thêm hoặc lưu thông tin chỉnh sửa *voucher* (hay *event*), **VoucherProvider** hỗ trợ **VoucherDialog/EventDialog** bằng các phương thức tương ứng như *addVoucher* cho thao tác thêm, *editVoucher* cho thao tác chỉnh sửa.



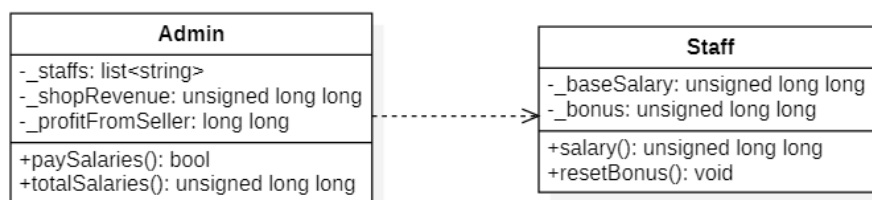
Hình III.5.9. Mối quan hệ giữa VoucherProvider và VoucherDialog, EventDialog

III.5.4. Giữa các lớp thành phần

Giữa các lớp thành phần với nhau, ta có *mối quan hệ phụ thuộc* giữa lớp **Admin** và lớp **Staff**.

Ta sẽ phân tích về 2 thao tác được lớp **Admin** hỗ trợ, bao gồm *thao tác tính tổng lương* và *trả lương*, ứng với các phương thức *totalSalaries* và *paySalaries* của lớp này. Cụ thể như sau:

- **Tính tổng lương:** chương trình thực hiện duyệt danh sách các nhân viên để gọi hàm *salary* trong lớp **Staff**, lấy ra thông tin về lương của mỗi nhân viên tại thời điểm đó.
- **Trả lương:** Sau khi gọi hàm tính tổng lương để xác định tổng số tiền phải trả, chương trình thực hiện thao tác trừ tiền vào ví *Bily* của quản lý một lượng tương ứng và cộng tiền vào trong ví *Bily* của mỗi nhân viên. Cuối cùng là cập nhật lại tiền thưởng cho mỗi nhân viên về 0, thông qua việc gọi đến phương thức *resetBonus* trong lớp **Staff**.

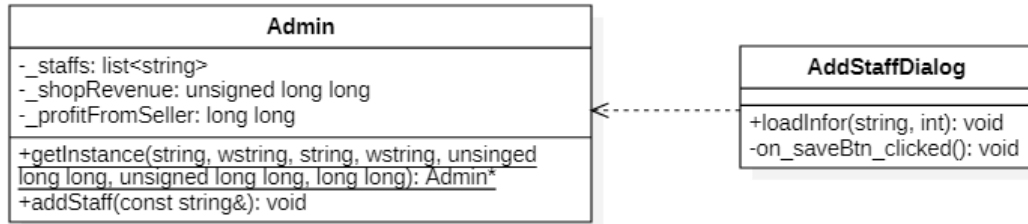


Hình III.5.10. Quan hệ dependency giữa lớp Admin và lớp Staff

III.5.5. Giữa các lớp thành phần và các lớp giao diện

Nói đến quan hệ giữa các lớp thành phần và các lớp giao diện, ta có *mối quan hệ phụ thuộc* giữa lớp **Admin** và lớp **AddStaffDialog**.

Để có thể hỗ trợ cho thao tác thêm một *người bán thành nhân viên*, lớp **AddStaffDialog** cần có sự hỗ trợ của của phương *addStaff* trong lớp **Admin**. Từ đó, lớp **AddStaffDialog** phụ thuộc vào lớp **Admin**.



Hình III.5.11. Quan hệ dependency giữa lớp *AddStaffDialog* và lớp *Admin*

IV. Môi trường phát triển

Ứng dụng **Bily Flower Shop** được nhóm em phát triển bằng những công nghệ sau:

- Ngôn ngữ lập trình **C++**, chuẩn **C++11**
- Sử dụng **Qt Framework** để lập trình giao diện với IDE **Qt Creator**
- Hệ điều hành **Window 10 64-bit**

Ứng dụng chạy ổn định nhất trên hệ điều hành **Window 10 64 – bit**. Trong trường hợp khởi động ứng dụng và gặp lỗi như thiếu các file **MSVCP140.dll**, **VCRUNTIME140.dll**, ta mở file **vc_redist.x64.exe** để cài đặt những file đó trước, sau đó mở lại file **BILY.exe** để khởi động ứng dụng.

Name	Date modified	Type	Size
Database	1/5/2022 9:30 PM	File Folder	
iconengines	1/8/2022 6:21 PM	File Folder	
imageformats	1/8/2022 6:21 PM	File Folder	
platforms	1/8/2022 6:21 PM	File Folder	
styles	1/8/2022 6:21 PM	File Folder	
translations	1/8/2022 6:21 PM	File Folder	
BILY.exe	1/8/2022 6:20 PM	Application	9,552 KB
D3Dcompiler_47.dll	3/11/2014 5:54 PM	Application extens...	4,077 KB
opengl32sw.dll	6/4/2020 2:50 PM	Application extens...	20,150 KB
Qt6Core.dll	6/7/2021 10:47 PM	Application extens...	5,336 KB
Qt6Gui.dll	6/7/2021 10:47 PM	Application extens...	7,241 KB
Qt6Svg.dll	6/7/2021 10:56 PM	Application extens...	347 KB
Qt6Widgets.dll	6/7/2021 10:47 PM	Application extens...	5,735 KB
vc_redist.x64.exe	11/13/2021 1:00 AM	Application	24,581 KB

Hình IV.1. Thư mục chứa ứng dụng với file thực thi **BILY.exe**

V. Tổng kết

Nhận xét về mặt ưu điểm, về cơ bản, nhóm chúng em đã thiết kế được hệ thống cho ứng dụng và hoàn thiện đầy đủ mục tiêu đã đặt ra, từ đó xây dựng được **Bily Flower Shop** tương đối ổn định với giao diện khá bắt mắt. Các tính chất của lập trình hướng đối tượng như tính đóng gói, kế thừa, đa hình đã được vận dụng khá hiệu quả. Bên cạnh đó, nhóm cũng đã nhìn nhận được tình huống phù hợp để áp dụng mẫu thiết kế *Singleton*.

Về khuyết điểm, những sự nhập nhằng trong mối quan hệ phụ thuộc (*dependency*) giữa các lớp chưa thực sự được nhóm giải quyết triệt để. Cụ thể, trong đồ án này, chúng em chỉ đánh giá rằng lớp A phụ thuộc vào lớp B khi A gọi đến một phương thức thực hiện một hành động cụ thể trong B (không phải là các phương thức *getter*). Bên cạnh đó, một số chức năng của ứng dụng như *voucher* và *event*, hay việc thêm một *người bán thành nhân viên* còn chưa thể hiện rõ được tính hợp lý trong thực tế.

Trong quá trình xây dựng **Bily Flower Shop**, chúng em đã học hỏi và rút ra được nhiều kinh nghiệm quý báu, về cả kiến thức chuyên môn lẫn kỹ năng làm việc nhóm. Cụ thể như biết cách xây dựng và phát triển một ứng dụng, từ ý tưởng sơ khai cho đến sản phẩm cuối cùng. Chúng em đã biết thêm về cách tổ chức và quản lý các lớp, các đối tượng cũng như những mối quan hệ giữa chúng một cách hợp lý, phát huy những tính chất ưu việt của lập trình hướng đối tượng. Đồng thời, cách làm việc nhóm cũng dần trở nên chuyên nghiệp hơn, từ khâu phân chia công việc cho đến các thời hạn hoàn thành công việc, luôn đảm bảo tốt tiến độ. Đó chắc chắn là những bước đệm quan trọng để chúng em xây dựng các phần mềm, ứng dụng lớn hơn sau này.

Trong tương lai, **Bily Flower Shop** có thể được cải tiến để nâng cao chất lượng dịch vụ, phù hợp với nhu cầu của người dùng. Đầu tiên là sự đầu tư về cơ sở dữ liệu thật sự, thay vì sử dụng các file có định dạng JSON như hiện tại. Sau đó, ứng dụng có thể mở rộng nhiều đối tượng tham gia hơn, đa dạng các nhân viên khác nhau của cửa hàng như: nhân viên thiết kế, nhân viên thu ngân, nhân viên giao hàng, ... Bên cạnh đó, ta có thể cho ra đời nhiều loại mã giảm giá hơn, như mã giảm phí vận chuyển, mã hoàn tiền vào ví điện tử, ... kết hợp với đa dạng hóa sản phẩm của cửa hàng như thêm các quà tặng, phụ kiện đi kèm sản phẩm nhằm thu hút khách hàng. Ngoài ra, có thể thêm tính năng chat giữa những người dùng với nhau phục vụ nhu cầu trao đổi, cung cấp thông tin về sản phẩm họ quan tâm; tính năng đổi trả hàng nếu khách hàng không hài lòng với dịch vụ của Bily; tính năng chia sẻ để khách hàng có thể giới thiệu với bạn bè về những sản phẩm họ yêu thích; ... Chúng em tin rằng với những đề xuất trên, ứng dụng có thể ngày càng được mở rộng và phát triển hơn nữa, phục vụ tăng sức cạnh tranh trong thị trường.

VI. Tự đánh giá

VI.1. Bảng điểm

STT	Tiêu chí	Điểm	Đánh giá
1	Tài liệu mô tả phần mềm	0.5	0.5
2	Tài liệu thiết kế phần mềm: <ul style="list-style-type: none">- Phân tích đầy đủ các đối tượng- Vẽ sơ đồ UML	3	2.5
3	Ứng dụng được ít nhất 5 kỹ thuật đã học trong OOP	3	2.75
4	Giao diện đẹp	1	1
5	Độ hoàn thiện	1	1
6	Trình bày	1	0.75
7	Cơ sở dữ liệu	0.5	0.5
8	Ứng dụng được ít nhất một mẫu design pattern	1	1
	Tổng điểm	11	10

Bảng VI.1. Bảng điểm tự đánh giá

VI.2. Đóng góp của các thành viên

Ba thành viên cùng lên ý tưởng, thiết kế hệ thống, hỗ trợ nhau tìm, sửa lỗi và cùng viết báo cáo cho đồ án này. Lưu ý rằng, bảng phân công bên dưới chỉ mang tính chất tương đối vì có những phần công việc có sự giao thoa nhau và có sự đóng góp của nhiều thành viên.

Thành viên	Công việc	Phần trăm đóng góp
Hoàng Trọng Vũ 20120025 (nhóm trưởng)	Lớp AccountProvider, lớp Account và các lớp kế thừa	45%
	Lớp LoginProvider	
	Lập trình giao diện	
Nguyễn Thiên An 20120030	Thu thập dữ liệu	27,5%
	Lớp OrderProvider, Order	
	Thao tác thống kê dữ liệu	
	Lớp VoucherProvider, Voucher	
	Thiết kế hình ảnh, icon	
Dũ Quốc Huy 20120101	Thu thập dữ liệu	27,5%
	Vẽ sơ đồ UML	
	Lớp ProductProvider, Product	
	Thao tác tìm kiếm sản phẩm	

Bảng VI.2. Đóng góp của các thành viên

VII. Tài liệu tham khảo

- [1]. Trần Đan Thư, Đinh Bá Tiến, Nguyễn Tấn Trần Minh Khang, *Phương pháp lập trình hướng đối tượng*, Nhà xuất bản khoa học và kỹ thuật
- [2]. Slides bài giảng học phần *Phương pháp lập trình hướng đối tượng*, Thầy Nguyễn Minh Huy và Thầy Trần Anh Duy, Trường Đại học Khoa học Tự nhiên, ĐHQG TP.HCM
- [3]. Thư viện **JSON Modern for C++** từ <https://github.com/nlohmann/json>
- [4]. Mã hóa SHA256 từ <http://www.zedwood.com/article/cpp-sha256-function>
- [5]. Các hàm chuyển đổi chuỗi ký tự UTF16 sang UTF8 và ngược lại:
 - <https://cppdeveloper.com/c-nang-cao/ham-convert-chuoi-ky-tu-tu-utf16-sang-utf8-su-dung-c-native-code/>
 - <https://stackoverflow.com/questions/7153935/how-to-convert-utf-8-stdstring-to-utf-16-stdwstring/38383389>
- [6]. Qt document: <https://doc.qt.io/>
- [7]. Mẫu thiết kế Singleton: <https://cppdeveloper.com/design-patterns/design-patterns-1-singleton-pattern/>
- [8]. Dữ liệu về các sản phẩm được thu thập từ <http://hoayeuthuong.com>