TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ MÔN MẪU THIẾT KẾ**

**SHOP BÁN ĐỒ THỜI TRANG**

*Người hướng dẫn*: **THẦY VŨ ĐÌNH HỒNG**

*Người thực hiện*: **LƯU KHANG HUY– 52100805**

**LÊ THÀNH PHÚC– 52100833**

Lớp **: 21050261**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ MÔN MẪU THIẾT KẾ**

**SHOP BÁN ĐỒ THỜI TRANG**

Người hướng dẫn: **THẦY VŨ ĐÌNH HỒNG**

Người thực hiện: **LƯU KHANG HUY– 52100805**

**LÊ THÀNH PHÚC– 52100833**

Lớp **: 21050261**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến thầy Vũ Đình Hồng đã quan tâm, giúp đỡ tụi em trong môn học mẫu thiết kế này.

Tuy nhiên kiến thức trong môn học này quá nhiều nên trong quá trình làm đề tài chúng em sẽ không tránh được những thiếu sót. Chúng em mong nhận được những ý kiến đóng góp từ thầy để bài làm của chúng em được chỉnh chu là hoàn thiện hơn.

Chúng em chân thành cảm ơn thầy.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của thầy Vũ Đình Hồng;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 28 tháng 4 năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Lưu Khang Huy*

*Lê Thành Phúc*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Báo cáo này tập trung vào việc áp dụng một loạt các design pattern quan trọng để cải thiện cấu trúc và hiệu suất của mã nguồn trong quá trình phát triển phần mềm. Trong bài báo cáo này, chúng em áp dụng sử dụng các pattern sau: Factory Pattern, MVC Pattern, Connection Pool Pattern, Singleton.

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc165405823)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 3](#_Toc165405824)

[TÓM TẮT 4](#_Toc165405825)

[MỤC LỤC 5](#_Toc165405826)

[MỞ ĐẦU 6](#_Toc165405827)

[CHƯƠNG 1: ÁP DỤNG MVC PATTERN 8](#_Toc165405828)

[**1.1** **Giới thiệu về MVC Pattern** 8](#_Toc165405829)

[**1.2** **Áp dụng cho dự án** 8](#_Toc165405830)

[**1.3** **Lợi ích của MVC Pattern** 9](#_Toc165405831)

[CHƯƠNG 2: ÁP DỤNG CONNECTION POOL PATTERN 10](#_Toc165405832)

[2.1 Giới thiệu về Connection Pool 10](#_Toc165405833)

[**2.2 Áp dụng Connection Pool trong dự án** 10](#_Toc165405834)

[**2.3 Lợi ích của việc áp dụng Connection Pool** 11](#_Toc165405835)

[CHƯƠNG 3: ÁP DỤNG SINGLETON PATTERN 11](#_Toc165405836)

[**3.1 Giới thiệu về Singleton Pattern** 12](#_Toc165405837)

[**3.2 Áp dụng Singleton Pattern trong dự án** 12](#_Toc165405838)

[3.3 Lợi ích của Singleton Pattern 12](#_Toc165405839)

[CHƯƠNG 4: ÁP DỤNG FACTORY PATTERN 13](#_Toc165405840)

[**4.1 Giới thiệu về Factory Pattern** 13](#_Toc165405841)

[**4.2 Áp dụng Factory Pattern trong dự án** 14](#_Toc165405842)

[**4.3 Lợi ích của Factory Pattern** 14](#_Toc165405843)

[TÀI LIỆU THAM KHẢO 16](#_Toc165405844)

MỞ ĐẦU

Trong quá trình phát triển phần mềm, việc áp dụng các design pattern đóng vai trò quan trọng trong việc tối ưu hóa cấu trúc và hiệu suất của mã nguồn. Các design pattern không chỉ cung cấp một cách tiếp cận cấu trúc hóa mã nguồn một cách logic và có tổ chức, mà còn giúp giải quyết các vấn đề phức tạp một cách hiệu quả và linh hoạt.

Trong báo cáo này, chúng em tập trung vào việc sử dụng các design pattern quan trọng: Factory, MVC (Model-View-Controller), Object Pool và Singleton. Mỗi design pattern này được áp dụng để giải quyết các vấn đề cụ thể trong quá trình phát triển và cải thiện tính module, tính tái sử dụng và tính bảo trì của hệ thống.

Chúng em sẽ đi vào chi tiết về cách chúng em đã sử dụng mỗi design pattern, cùng với những lợi ích mà nó mang lại cho dự án. Từ việc tạo đối tượng linh hoạt đến việc tổ chức mã nguồn theo mô hình MVC, từ quản lý tài nguyên đến việc đảm bảo sự duy nhất của một đối tượng, mỗi design pattern đóng góp vào việc xây dựng một hệ thống phần mềm mạnh mẽ và dễ bảo trì.

Tiếp tục đọc để khám phá cách mà việc sử dụng các design pattern đã cải thiện quá trình phát triển và chất lượng của mã nguồn trong dự án của chúng em.

CHƯƠNG 1: ÁP DỤNG MVC PATTERN

Trong chương này, chúng em sẽ tìm hiểu về việc áp dụng MVC (Model-View-Controller) Pattern trong dự án của chúng em. MVC là một mô hình phổ biến được sử dụng để tổ chức mã nguồn trong các ứng dụng phần mềm, tách biệt logic xử lý dữ liệu, giao diện người dùng và điều khiển luồng của ứng dụng thành các phần riêng biệt.

* 1. **Giới thiệu về MVC Pattern**

MVC Pattern bao gồm ba thành phần chính:

* *Model*: Đại diện cho dữ liệu và logic xử lý. Model có trách nhiệm quản lý và cung cấp dữ liệu cho ứng dụng.
* *View*: Đại diện cho giao diện người dùng. View hiển thị dữ liệu từ Model cho người dùng và thu thập dữ liệu đầu vào từ người dùng.
* *Controller*: Điều khiển luồng của ứng dụng. Controller nhận các yêu cầu từ người dùng thông qua View, tương tác với Model để lấy hoặc cập nhật dữ liệu, và sau đó cập nhật lại giao diện người dùng thông qua View.
  1. **Áp dụng cho dự án**

Trong dự án của chúng em, chúng em đã áp dụng MVC Pattern để tách biệt logic xử lý dữ liệu, giao diện người dùng và điều khiển luồng của ứng dụng một cách rõ ràng và có tổ chức. Cụ thể:

* Model: Chúng em đã xây dựng các lớp Model để đại diện cho dữ liệu và logic xử lý trong ứng dụng. Các lớp Model này thực hiện các thao tác như truy xuất dữ liệu từ cơ sở dữ liệu, xử lý logic kinh doanh, và cung cấp dữ liệu cho Controller.
* View: Chúng em đã thiết kế các giao diện người dùng dựa trên các yêu cầu của khách hàng và thiết kế tổ chức của MVC Pattern. Các View này hiển thị thông tin từ Model cho người dùng và thu thập thông tin đầu vào từ người dùng để truyền cho Controller xử lý.
* Controller: Chúng em đã triển khai các Controller để điều khiển luồng của ứng dụng. Các Controller này nhận các yêu cầu từ người dùng thông qua View, tương tác với Model để lấy hoặc cập nhật dữ liệu, và sau đó cập nhật lại giao diện người dùng thông qua View.
  1. **Lợi ích của MVC Pattern**
* Tách biệt mối quan tâm: MVC Pattern giúp tách biệt các mối quan tâm của ứng dụng, làm cho việc phát triển và bảo trì dễ dàng hơn. Các nhà phát triển có thể tập trung vào việc viết mã cho từng thành phần mà không cần phải lo lắng về cách thức các thành phần tương tác với nhau.
* Tái sử dụng mã: MVC Pattern thúc đẩy sự tái sử dụng mã. Các thành phần MVC có thể được sử dụng lại trong các ứng dụng khác nhau.
* Khả năng mở rộng: MVC Pattern giúp ứng dụng dễ dàng mở rộng. Các thành phần MVC có thể được thêm vào hoặc thay đổi mà không ảnh hưởng đến các thành phần khác.

Trong các phần tiếp theo của báo cáo, chúng em sẽ tiếp tục giới thiệu về việc áp dụng các design pattern khác trong dự án của mình và những lợi ích mà chúng mang lại.

CHƯƠNG 2: ÁP DỤNG CONNECTION POOL PATTERN

Trong chương này, chúng em sẽ nói về việc áp dụng Connection Pool Pattern trong dự án của chúng em. Connection Pool Pattern là một kỹ thuật quan trọng được sử dụng để quản lý và tái sử dụng các kết nối đến cơ sở dữ liệu, giúp cải thiện hiệu suất của ứng dụng và giảm tải cho cơ sở dữ liệu.

## 2.1 Giới thiệu về Connection Pool

Connection Pool là một bộ sưu tập các kết nối đến cơ sở dữ liệu đã được tạo trước và duy trì trong suốt thời gian hoạt động của ứng dụng. Thay vì mở và đóng kết nối mới mỗi khi cần truy cập cơ sở dữ liệu, ứng dụng có thể tái sử dụng các kết nối có sẵn trong Connection Pool. Điều này giúp giảm thiểu thời gian tạo và đóng kết nối, giảm tải cho cơ sở dữ liệu và tăng hiệu suất ứng dụng.

**2.2 Áp dụng Connection Pool trong dự án**

Trong trang web bán đồ thời trang, Connection Pool Pattern được áp dụng như sau:

* Connection Pool: Connection pool có thể được sử dụng để lưu trữ các kết nối cơ sở dữ liệu được sử dụng bởi ứng dụng.
* Connection Factory: Connection factory có thể được sử dụng để tạo các kết nối cơ sở dữ liệu mới khi cần thiết.
* Connection Manager: Connection manager có thể được sử dụng để quản lý tập hợp các kết nối trong pool, bao gồm việc lấy và trả về các kết nối, theo dõi trạng thái của các kết nối và loại bỏ các kết nối không còn sử dụng.hình và để xử lý các trường hợp ngoại lệ như kết nối đến cơ sở dữ liệu bị ngắt kết nối.

**2.3 Lợi ích của việc áp dụng Connection Pool**

* Cải thiện hiệu suất: Connection Pool Pattern giúp cải thiện hiệu suất của ứng dụng bằng cách giảm thiểu số lượng kết nối cơ sở dữ liệu mới cần được tạo. Việc tạo kết nối cơ sở dữ liệu có thể tốn thời gian, vì vậy việc tái sử dụng các kết nối hiện có có thể giúp cải thiện thời gian phản hồi của ứng dụng.
* Tăng khả năng mở rộng: Connection Pool Pattern giúp tăng khả năng mở rộng của ứng dụng bằng cách cho phép ứng dụng xử lý nhiều yêu cầu đồng thời hơn. Connection pool có thể được cấu hình để chứa một số lượng kết nối nhất định, và ứng dụng sẽ chờ đợi khi tất cả các kết nối đều đang được sử dụng.
* Giảm chi phí: Connection Pool Pattern giúp giảm chi phí bằng cách giảm thiểu số lượng kết nối cơ sở dữ liệu cần được duy trì. Việc duy trì các kết nối cơ sở dữ liệu có thể tốn kém, vì vậy việc tái sử dụng các kết nối hiện có có thể giúp tiết kiệm tiền.

CHƯƠNG 3: ÁP DỤNG SINGLETON PATTERN

Trong chương này, chúng em sẽ đề cập đến việc áp dụng Singleton Pattern trong dự án của chúng em. Singleton Pattern là một trong những design pattern cơ bản nhất, được sử dụng để đảm bảo rằng chỉ có một phiên bản duy nhất của một lớp được tạo ra và cung cấp một cách tiếp cận toàn cục đến nó.

**3.1 Giới thiệu về Singleton Pattern**

Singleton Pattern là một mô hình thiết kế phần mềm đảm bảo rằng chỉ có một thể hiện duy nhất của một lớp được tạo trong suốt quá trình chạy của chương trình. Mô hình này hữu ích khi bạn cần một đối tượng duy nhất để truy cập các tài nguyên chung hoặc cung cấp một điểm truy cập toàn cầu cho một chức năng.

Singleton Pattern bao gồm các thành phần sau:

* Singleton Class: Lớp mà bạn muốn chỉ có một thể hiện.
* GetInstance Method: Phương thức để lấy thể hiện duy nhất của lớp.

**3.2 Áp dụng Singleton Pattern trong dự án**

* Quản Lý Cấu Hình Toàn Cục: Chúng em đã sử dụng Singleton Pattern để quản lý cấu hình toàn cục của ứng dụng. Điều này giúp chúng em truy cập cấu hình từ bất kỳ nơi nào trong mã nguồn một cách dễ dàng mà không cần tạo ra nhiều thể hiện của đối tượng cấu hình.
* Quản Lý Kết Nối Đến Cơ Sở Dữ Liệu: Chúng em cũng đã sử dụng Singleton Pattern để quản lý các kết nối đến cơ sở dữ liệu. Điều này đảm bảo rằng chỉ có một kết nối đến cơ sở dữ liệu được sử dụng trong suốt quá trình thực thi của ứng dụng, giúp tối ưu hóa tài nguyên và quản lý tình trạng kết nối một cách hiệu quả.

3.3 Lợi ích của Singleton Pattern

* Đảm Bảo Duy Nhất: Singleton Pattern đảm bảo rằng chỉ có một phiên bản duy nhất của một lớp được tạo ra, đảm bảo tính nhất quán và đồng nhất trong toàn bộ ứng dụng.
* Dễ Dàng Tiếp Cận Toàn Cục: Singleton Pattern cung cấp một cách tiếp cận toàn cục đến thể hiện duy nhất của lớp, giúp quản lý và truy cập đối tượng một cách dễ dàng từ bất kỳ nơi nào trong ứng dụng.
* Tối Ưu Hóa Tài Nguyên: Singleton Pattern giúp tối ưu hóa việc sử dụng tài nguyên bằng cách giảm thiểu số lượng đối tượng được tạo ra, đặc biệt là đối với các tài nguyên có chi phí khởi tạo cao như kết nối đến cơ sở dữ liệu.

Trong các phần tiếp theo của báo cáo, chúng em sẽ tiếp tục giới thiệu về việc áp dụng các design pattern khác và những lợi ích mà chúng mang lại cho dự án của chúng em.

CHƯƠNG 4: ÁP DỤNG FACTORY PATTERN

Trong chương này, chúng em sẽ tìm hiểu về việc áp dụng Factory Pattern trong dự án của chúng em. Factory Pattern là một design pattern được sử dụng để tạo ra đối tượng mà không cần chỉ định cụ thể lớp cụ thể của chúng, tạo điều kiện cho tính linh hoạt và dễ dàng mở rộng của mã nguồn.

**4.1 Giới thiệu về Factory Pattern**

Factory Pattern là một mô hình thiết kế phần mềm được sử dụng để tạo các đối tượng của một lớp cụ thể. Mô hình này giúp tách biệt việc tạo đối tượng khỏi việc sử dụng đối tượng, giúp cho mã dễ dàng bảo trì và mở rộng hơn.

Factory Pattern bao gồm các thành phần sau:

* Factory Interface: Giao diện định nghĩa phương thức để tạo các đối tượng.
* Concrete Factory Classes: Các lớp cụ thể triển khai giao diện factory và tạo các đối tượng của lớp cụ thể.
* Client Code: Mã sử dụng factory để tạo các đối tượng mà không cần biết lớp cụ thể nào được tạo.

**4.2 Áp dụng Factory Pattern trong dự án**

* Tạo Đối Tượng Cấu Hình: Chúng em đã sử dụng Factory Pattern để tạo đối tượng cấu hình của ứng dụng. Factory Pattern cho phép chúng em tạo đối tượng cấu hình mà không cần biết cụ thể lớp cấu hình được sử dụng, giúp tạo ra tính linh hoạt và dễ dàng mở rộng.
* Xử Lý Đối Tượng Tùy Chỉnh: Factory Pattern đã được sử dụng để xử lý việc tạo ra các đối tượng tùy chỉnh theo yêu cầu của người dùng. Thay vì mỗi lần cần tạo đối tượng mới, chúng em đã sử dụng Factory Pattern để tạo ra các đối tượng tùy chỉnh một cách linh hoạt và dễ dàng.

**4.3 Lợi ích của Factory Pattern**

* Tính Linh Hoạt và Mở Rộng: Factory Pattern giúp tạo ra mã nguồn linh hoạt và dễ dàng mở rộng bằng cách tách biệt quá trình tạo đối tượng từ mã nguồn sử dụng đối tượng.
* Giảm Độ Phụ Thuộc: Factory Pattern giúp giảm độ phụ thuộc giữa các lớp trong mã nguồn, vì mã nguồn không cần biết cụ thể lớp cụ thể của đối tượng mà nó tạo ra.
* Tái Sử Dụng Mã Nguồn: Factory Pattern giúp tạo điều kiện cho việc tái sử dụng mã nguồn bằng cách cung cấp một cơ chế chung để tạo ra các đối tượng.

CHƯƠNG 5: ÁP DỤNG OBSERVER PATTERN

**5.1 Giới thiệu về Observer Pattern**

Observer Pattern bao gồm hai loại đối tượng chính:

* Subject: Là đối tượng mà các Observer quan sát. Subject chứa danh sách các Observer và cung cấp các phương thức để thêm, xóa và thông báo cho các Observer khi trạng thái của nó thay đổi.
* Observer: Là các đối tượng quan sát Subject và nhận thông báo khi Subject thay đổi trạng thái. Observer cung cấp một phương thức để cập nhật trạng thái của nó dựa trên trạng thái mới của Subject.

**5.2 Áp dụng Observer trong dự án**

* Quản Lý Sự Kiện: Chúng em đã sử dụng Observer Pattern để quản lý sự kiện trong ứng dụng. Khi một sự kiện xảy ra, Subject thông báo cho tất cả các Observer đã đăng ký và cập nhật trạng thái của chúng dựa trên sự kiện.

**5.3 Lợi ích của Observer Pattern**

* Tính Linh Hoạt và Tính Mở Rộng: Observer Pattern giúp tạo ra mã nguồn linh hoạt và dễ dàng mở rộng bằng cách tách biệt Subject và Observer.
* Giảm Độ Phụ Thuộc: Observer Pattern giúp giảm độ phụ thuộc giữa các đối tượng trong mã nguồn, vì các Observer không cần biết cụ thể Subject mà nó quan sát.
* Tái Sử Dụng Mã Nguồn: Observer Pattern giúp tạo điều kiện cho việc tái sử dụng mã nguồn bằng cách cung cấp một cơ chế chung để quản lý sự kiện và cập nhật trạng thái.

CHƯƠNG 6: ÁP DỤNG STRATEGY PATTERN

**6.1 Giới thiệu về Strategy Pattern**

Trong Strategy Pattern, chúng ta có ba thành phần chính:

* Context: Là đối tượng chứa một thuật toán cụ thể và có khả năng chuyển đổi giữa các thuật toán.
* Strategy: Là interface hoặc lớp trừu tượng đại diện cho một thuật toán cụ thể.
* Concrete Strategies: Là các lớp cụ thể triển khai các thuật toán cụ thể.

**6.2 Áp dụng Strategy trong dự án**

- Quản lý mật khẩu đăng ký: Chúng em đã sử dụng StrategyPattern để quản lý mật khảu khi người dùng đăng ký.

**6.3 Lợi ích của Strategy Pattern**

* Tính Linh Hoạt và Tính Mở Rộng: Strategy Pattern giúp chúng ta tạo ra mã nguồn linh hoạt và dễ dàng mở rộng bằng cách đóng gói các thuật toán vào các đối tượng riêng biệt.
* Tách Biệt Quyết Định và Hành Động: Strategy Pattern tách biệt quyết định (chọn lựa thuật toán) và hành động (thực thi thuật toán), làm cho mã nguồn trở nên dễ đọc và dễ bảo trì hơn.
* Tái Sử Dụng Mã Nguồn: Strategy Pattern giúp tạo điều kiện cho việc tái sử dụng mã nguồn bằng cách cung cấp một cơ chế chung để quản lý các thuật toán.

CHƯƠNG 7: ÁP DỤNG PAGINATION PATTERN

**7.1 Giới thiệu về Pagination Pattern**

Trong **Pagination** Pattern được xử dụng để phân trang sản phẩm. Mặc dù là Design pattern không chính thức nhưng rất cần thiết cho các website

**7.2 Áp dụng Pagination trong dự án**

- Phân trang: sử dụng phân trang để chia các sản phẩm theo từng trang để quản lý số lượng sản phẩm.

**7.3 Lợi ích của Pagination Pattern**

* Hiệu suất tải trang: Khi làm việc với các tập dữ liệu lớn, phân trang giúp giảm số lượng dữ liệu cần tải lên một lần. Thay vì hiển thị toàn bộ dữ liệu trong một trang, chỉ một phần nhỏ được hiển thị ban đầu. Điều này giúp giảm thời gian tải trang và tăng hiệu suất chung của ứng dụng.
* Tính tương tác người dùng tốt hơn: Phân trang cho phép người dùng dễ dàng điều hướng qua các trang dữ liệu khác nhau. Người dùng có thể chọn trang tiếp theo, trang trước đó hoặc nhảy trực tiếp đến một trang cụ thể. Điều này cung cấp một trải nghiệm tương tác tốt hơn và cho phép người dùng khám phá dữ liệu một cách linh hoạt.

TÀI LIỆU THAM KHẢO