STATIC PERSPECTIVE

Decomposition style and Layer style and Uses style  
  
**1. Primary presentation:**



1. **Element catalog:**
2. Elements and their properties

|  |  |  |
| --- | --- | --- |
| Elements | | Properties |
| GUI | ProductUI | Thể hiện giao diện để người quản lý có thể kiểm soát và thực hiện các thao tác lên giao diện như thêm mới, cập nhật, xoá sản phẩm. |
| CategoryUI | Thể hiện giao diện quản lý các danh mục sản phẩm để người quản lý có thể thêm mới, cập nhật, xoá sản phẩm vào danh mục. Người quản lý có thể cập nhật, thêm mới hoặc xoá một danh mục khác. |
| SynchUI | Đây là giao diện admin có thể chỉnh sửa chế độ đồng bộ của hệ thống:   * Đồng bộ bằng tay: là quá trình đồng bộ do người cashier thực hiện mỗi khi cần thiết * Đồng bộ tự động: người admin sẽ quy định thời gian đồng bộ dữ liệu, và cứ đến giờ quy định thì hệ thống sẽ đồng bộ. |
| UserAccount | Đây là giao diện quản lý tài khoản của user. Người admin có thể thêm mới, chỉnh sửa hay huỷ bỏ quyền của user. |
| StatisticUI | Giao diện này giúp người quản lý có thể thống kê sản phẩm như: thống kê những sản phẩm bán ra trong ngày/trong tuần/trong tháng/ trong năm. |
| SalesUI | Giao diện này giúp người cashier có thể thực hiện việc thanh toán bill cho khách hàng. Lựa chọn phương thức thanh toán tiền với khách hàng như: thanh toán bằng điểm hay trả tiền mặt. |
| Business Logic | ProductBL | Thực hiện các công việc xử lý tính toán mà người user thực hiện trên giao diện ProductUI | |
| CategoryBL | Thực hiện các công việc xử lý tính toán mà người user thực hiện trên giao diện CategoryUI | |
| StatisticBL | Thực hiện các công việc xử lý tính toán mà người user thực hiện trên giao diện StatisticUI | |
| UserAccountBL | Thực hiện các công việc xử lý tính toán mà người user thực hiện trên giao diện MagementUI | |
| SaleBL | Thực hiện các công việc xử lý tính toán mà người user thực hiện trên giao diện SalesUI | |
| Common | Đây là class chứa các hàm (hay biến) dùng chung cho các class khác ở layer Business Logic | |
| Data Transfer Object | ProductDTO | Chưa các biến thuộc riêng về Product (VD: mã Product, Tên product, giá tiền, mã danh mục), không chứa các hàm khác. | |
| CategoryDTO | Chưa các biến thuộc riêng về Category (VD: mã Category, tên Category), không chứa các hàm khác. | |
| CustomerDTO | Chưa các biến thuộc riêng về Customer (VD: mã Customer, tên Customer, số điện thoại, địa chỉ, tổng điểm), không chứa các hàm khác. | |
| BillDTO | Chưa các biến thuộc riêng về việc thanh toán bill với khách hàng (VD: mã Bill, ngày thực hiện,…), không chứa các hàm khác. | |
| BillDetailDTO | Chứa các biến thể hiện thông tin chi tiết của Bill (), không chứa các hàm khác. | |
| UserAccountDTO | Chưa các biến thuộc riêng về tài khoảng người dùng (VD: UserID, password,…), không chứa các hàm khác. | |
| Data Access | UserAccountDB | Chứa các hàm xử lý việc truy xuất và lưu dữ liệu trên database liên quan đến tài khoảng người dùng | |
| ProductDB | Chứa các hàm xử lý việc truy xuất và lưu dữ liệu trên database liên quan đến sản phẩm | |
| CustomerDB | Chứa các hàm xử lý việc truy xuất và lưu dữ liệu trên database liên quan đến khách hàng | |
| CategoryDB | Chứa các hàm xử lý việc truy xuất và lưu dữ liệu trên database liên quan đến danh mục | |
| SaleDB | Chứa các hàm xử lý việc truy xuất và lưu dữ liệu trên database liên quan đến việc thanh toán hoá đơn. | |

1. Relations and their properties

|  |  |
| --- | --- |
| Connector | Properties |
| Allowed to use | Cho phép các class trong một layer này có thể sử dụng các class trong layer khác |

1. **Context diagram:**



1. **Architecture background:**

Hệ thống được chia làm 4 tier bao gồm: GUI layer, Business Logic Layer, Transfer Data Object Layer, Data Access Layer.

Khi phát triển mô hình 4 lớp sẽ giảm được thời gian bảo trì và phát triển chương trình vì chia theo mô hình đa lớp tức là giống với việc sử dụng nguyên tắc chia để trị, khi thay đổi mộ hàm nào đó ở lớp này thì nó không ảnh hưởng nhiều đến lớp khác vì vậy chi phí bảo trì thấp hơn; có thể nâng cấp hoặc sửa đổi dễ dàng hơn. Từng lớp có thể được tái sử dụng lại nhiều lần ở nhiều ứng dụng khác nhau.

Mỗi lớp có thể được chạy trên các OS khác nhau ví dụ: database có thể chạy trên UNIX, LINUX; còn GUI layer thì chạy trên Windows (hoặc Web server) và được phát triển bằng nhiều ngôn ngữ lập trình cũng như nhiều nhóm phát triển.

Giới thiệu từng lớp và mục đích sử dụng:

* GUI layer: Lớp này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả/dữ liệu thông qua các thành phần trong giao diện người sử dụng. Lớp này sẽ sử dụng các dịch vụ do lớp Business Logic cung cấp. Theo phân tích yêu cầu đề bài thì lớp GUI gồm những class giao diện sau: ProductUI, CustomerUI, CategoryUI, StatisticUI, ManagementUI, SaleUI.
* Business Logic Layer: xử lý chính các dữ liệu trước khi được đưa lên hiển thị trên màn hình hoặc xử lý các dữ liệu trước khi lưu dữ liệu xuống cơ sở dữ liệu. Lớp business logic có thể kiểm tra và thực hiện có yêu cầu nghiệp vụ, các tính năng tính toán của chương trình sẽ thực thi theo một trình tự logic. Lớp này bao gồm các class sau: ProductBL, CustomerBL, CategoryBL, SaleRetailStoreBL, StatisticBL, UserAccountBL, Common.
* Data Access Layer: lớp này có nhiệm vụ xử lý các yêu cầu thao tác lên dữ liệu như cập nhật CSDL, đọc cơ sở dự liệu và trả về cho lớp Business Logic. Lớp này bao gồm các class như: UserDB, ProductDB, CustomerDB, CategoryDB, SaleRetailStoreDB.
* Data Transfer Object: Lớp này sẽ tạo ra các lớp object để hỗ trợ cho Data Business Logic Layer xử lý và sử dụng trong việc truyền tham số xuống lớp Data Access. Lớp này bao gồm các class như: ProductDTO, CategoryDTO, CustomerDTO, SaleRetailStoreDTO, UserDTO,.

1. **Glossary of terms:**
2. **Other information:**