**Học viện Công nghệ Bưu chính Viễn thông- Cơ sở TP. Hồ Chí Minh**

**KHOA : CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỊNH KỲ LẦN 1**

Đề tài : Xây dựng Web Quản lý Khách sạn

Giảng viên hướng dẫn: Nguyễn Ngọc Duy

Sinh viên thực tập: Trần Thị Loan

Mã sv: N13DCCN105

Mục tiêu đề tài: Xây dựng chương trình quản lý Khách Sạn

* Quản lý nhân viên
* Ghi nhận thông tin khách hàng thuê phòng
* Ghi nhận đăng ký đặt phòng
* Ghi nhận thuê phòng (check in)
* Tra cứu phòng qua sơ đồ phòng
* Ghi nhận việc sử dụng dịch vụ (minibar, giặt ủi, điện thoại, thức ăn, các dịch vụ khác)
* Tính tiền khi trả phòng – check out ( lập hóa đơn )
* Cập nhật giá phòng, giá dịch vụ
* Lập các báo cáo
  + - Nội dung thực hiện:

1. **Tìm hiểu về nghiệp vụ quản lý khách sạn**

Khảo sát:

* Nhân viên: mỗi nhân viên thuộc một bộ phận riêng (lễ tân, buồng, nhà hàng) phụ trách công việc nhất định. Một bộ phận có thể có nhiều nhân viên.
* Quy định đăng ký đặt phòng: khách sạn có nhiều loại phòng (Standard, Super, VIP…), kiểu phòng (1 giường đôi, 2 giường chiếc, 1 giường đôi và 1 giường chiếc…) ; Kết hợp loại và kiểu phòng ta có hạng phòng ; mỗi phòng sẽ thuộc 1 hạng phòng. Mỗi hạng phòng sẽ có giá nhất định từ 1 thời điểm xác định ; Chỉ nhận giữ chỗ nếu vào thời gian đó phòng còn trống và chưa được đặt chỗ trước. Phiếu đặt phòng ghi nhận thông tin khách đặt, khách thuê, số lượng phòng thuộc 1 hạng phòng được đặt, ngày bắt đầu thuê, số ngày thuê dự kiến ...
* Một phòng có thể có các trạng thái sau : sẵn sàng, có khách, dơ, đặt trước, bảo trì.
* Quy định thuê phòng: Có 2 loại phiếu thuê: phiếu thuê trên 1 phiếu đặt trước, hoặc phiếu thuê cho khách vãng lai (không đặt trước) ; Một phiếu thuê phòng quy định ngày đến, giờ đến, số phòng và thông tin các khách ở trong phòng đó. Lưu ý 1 phòng trong phiếu thuê có thể check in trễ hoặc check out sớm. Ta có thể giảm giá phòng cho nguyên đoàn hoặc 1 số phòng trong đoàn.
* Quy định sử dụng dịch vụ: khách sạn có phục vụ thêm 1 số lọai dịch vụ (ăn uống, giặt là, điện thoại, gửi fax, ... ) với đơn giá riêng cho từng loại. Riêng giặt ủi có 3 giá : giá giặt, giá ủi, giá hấp.
* Quy định tính tiền thuê phòng: mỗi hạng phòng có đơn giá riêng, giá phòng không có giá giờ, không phân biệt ngày lễ ngày thường. Giá phòng có thể giảm do khuyến mãi trong 1 khoảng thời gian.
* Thông tin khách hàng : ghi nhận thông tin liên lạc của khách hàng khi lần đầu tiên khách hàng đặt phòng hoặc thuê phòng. Nếu những thông tin của khách hàng có thay đổi, sẽ cập nhật thông tin khi khách thuê phòng.
* Người sử dụng có thể thay đổi các quy định sau: thay đổi số lượng khách tối đa, đơn giá các loại phòng, tiện nghi của phòng, danh mục dịch vụ, đơn giá các loại dịch vụ.
* Người quản trị hệ thống có thể thực hiện thao tác thêm, xóa, phân quyền theo từng bộ phận cho nhân viên khi đăng nhập vào hệ thống.
  + 1. Quy trình xử lý:

**Lập phiếu thuê phòng**

Khi khách đến khách sạn:

+ Đối với khách đã đặt phòng từ trước: khi yêu cầu nhận phòng phải đưa thông tin liên quan tới cuộc đặt phòng của mình. Nhân viên lễ tân sẽ tìm kiếm thông tin đặt phòng trên máy tính, xác nhận lại các thông tin và lưu vào cơ sở dữ liệu. Thông tin bao gồm thông tin về khách và thông tin phòng khách đặt.

+ Còn với khách chưa đặt phòng trước: có yêu cầu thuê phòng, nhân viên lễ tân sẽ cung cấp thông tin về các phòng trống hiện có cho khách thông qua việc truy vấn vào kho dữ liệu danh sách phòng. Khách xác nhận thuê phòng, nhân viên lễ tân sẽ lưu thông tin vào cơ sở dữ liệu.

**ĐÓN TIẾP**

**XÁC NHẬN VIỆC**

**ĐẶT PHÒNG**

**LÀM THỦ TỤC**

**ĐĂNG KÝ THUÊ PHÒNG**

**Đã đặt chỗ trước**

**TRA CỨU TÌNH TRẠNG PHÒNG**

**Chưa đặt chỗ trước**

**LẬP PHIẾU THUÊ PHÒNG**

**Có phòng trống**

***Quy trình phục vụ khách hàng trong thời gian khách lưu trú tại khách sạn:***

Trong thời gian khách lưu trú tại khách sạn, khi khách có yêu cầu sử dụng dịch vụ, khách sạn sẽ cung cấp các dịch vụ phục vụ nhu cầu của khách. Bộ phận lễ tân thu thập các phiếu sử dụng của khách hàng để tiến hành lập và tổng hợp lại các khoản chi phí của khách.

Sơ đồ quy trình phục vụ khách

**NHẬN YÊU CẦU SỬ DỤNG DỊCH VỤ**

**CUNG CẤP DỊCH VỤ**

**LƯU THÔNG TIN SỬ DỤNG DỊCH VỤ**

***Quy trình làm thủ tục và thanh toán cho khách:***

Khi khách có yêu cầu trả phòng, bộ phận lễ tân có nhiệm vụ thông báo cho các bộ phận khác trong khách sạn về việc trả phòng của khách nhằm xác định các khoản chưa thanh toán. Khi khách trả phòng, nhân viên lập hoá đơn tổng hợp và tiến hành thanh toán với khách.

Cuối mỗi ngày bộ phận lễ tân sẽ viết báo cáo tổng hợp về tình hình khách đến, khách đi, thực trạng phòng trống, phòng bận cũng như thu chi của ngày cũng như ý kiến phản hồi của khách… cho ban giám đốc khách sạn.

Sơ đồ quy trình làm thủ tục và thanh toán cho khách:

**NHẬN YÊU CẦU**

**TRẢ PHÒNG**

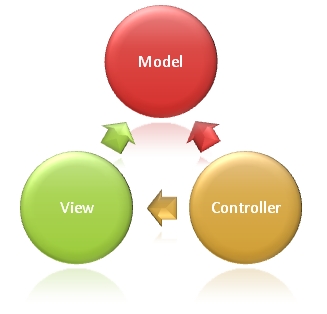
**XÁC NHẬN THÔNG TIN**

**TRẢ PHÒNG**

**LƯU THÔNG TIN TRẢ PHÒNG**

**TÍNH TIỀN PHÒNG VÀ TIỀN DỊCH VỤ**

1. **Tìm hiểu mô hình MVC asp.net**

***Tổng quan về ASP.NET MVC***  
Mẫu kiến trúc Model – View – Controller được sử dụng nhằm chi ứng dụng thành ba thành phần chính: model, view và controller. Nền tảng ASP.NET MVC giúp cho chúng ta có thể tạo được các ứng dụng web áp dụng mô hình MVC thay vì tạo ứng dụng theo mẫu ASP.NET Web Forsm. Nền tảng ASP.NET MVC có đặc điểm nổi bật là nhẹ (lighweigt), dễ kiểm thử phần giao diện (so với ứng dụng Web Forms), tích hợp các tính năng có sẵn của ASP.NET. Nền tảng ASP.NET MVC được định nghĩa trong namespace System.Web.Mvc và là một phần của name space System.Web.  
  
MVC là một mẫu thiết kế (design pattern) chuẩn mà nhiều lập trình viên đã quen thuộc. Một số loại ứng dụng web sẽ thích hợp với kiến trúc MVC. Một số khác vẫn thích hợp với ASP.NET Web Forms và cơ chế postbacks. Đôi khi có những ứng dụng kết hợp cả hai kiến trúc trên.  
  
Nền tảng MVC bao gồm các thành phần dưới đây:  
  
  
  
*Hình 01: Mẫu Model – View – Controller*  
*Models*: Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập logic của phần dữ liệu của ứng dụng. Thông thường, các đối tượng model lấy và lưu trạng thái của model trong CSDL. Ví dụ như, một đối tượng Product (sản phẩm) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng Products ở SQL Server.  
  
Trong các ứng dụng nhỏ, model thường là chỉ là một khái niệm nhằm phân biệt hơn là được cài đặt thực thụ, ví dụ, nếu ứng dụng chỉ đọc dữ liệu từ CSDL và gởi chúng đến view, ứng dụng khong cần phải có tầng model và các lớp lien quan. Trong trường hợp này, dữ liệu được lấy như là một đối tượng model (hơn là tầng model).  
  
*Views*: Views là các thành phần dùng để hiển thị giao diện người dùng (UI). Thông thường, view được tạo dựa vào thông tin dữ liệu model. Ví dụ như, view dùng để cập nhật bảng Products sẽ hiển thị các hộp văn bản, drop-down list, và các check box dựa trên trạng thái hiện tại của một đối tượng Product.  
  
*Controllers*: Controller là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng. Trong một ứng dụng MVC, view chỉ được dùng để hiển thị thông tin, controller chịu trách nhiệm quản lý và đáp trả nội dung người dùng nhập và tương tác với người dùng. Ví dụ, controller sẽ quản lý các dữ liệu người dùng gởi lên (query-string values) và gởi các giá trị đó đến model, model sẽ lấy dữ liệu từ CSDL nhờ vào các giá trị này.   
  
Mẫu MVC giúp bạn tạo được các ứng dụng mà chúng phân tách rạch ròi các khía cạnh của ứng dụng (logic về nhập liệu, logic xử lý tác vụ và logic về giao diện). Mẫu MVC chỉ ra mỗi loại logic kể trên nên được thiếp lập ở đâu trên ứng dụng. Logic giao diện (UI logic) thuộc về views. Logic nhập liệu (input logic) thuộc về controller. Và logic tác vụ (Business logic – là logic xử lý thông tin, mục đích chính của ứng dụng) thuộc về model. Sự phân chia này giúp bạn giảm bớt được sự phức tạp của ứng dụng và chỉ tập trung vào mỗi khía cạnh cần được cài đặt ở mỗi thời điểm. Ví dụ như bạn chỉ cần tập trung vào giao diện (views) mà không phải quan tâm đến logic xử lý thông tin của ứng dụng.  
  
Để quản lý sự phức tạp của ứng dụng, mẫu MVC giúp cho chúng ta có thể kiểm thử ứng dụng dễ dàng hơn hẳn so với khi áp dụng mẫu Web Forms. Ví dụ, trong một ứng dụng ASP.NET Web Forms, một lớp thường được sử dụng để hiển thị thông tin xuất ra cho người dùng và đồng thời xử lý thông tin người dùng nhập. Việc xây dựng các bộ test tự động cho ứng dụng Web Forms là rất phức tạp, bởi để kiểm thử mỗi trang web, bạn phải khởi tạo đối tượng trang, khởi tạo tất cả các control được sử dụng trong trang và các lớp phụ thuộc trong ứng dụng. Và bởi vì có quá nhiều lớp cần được khởi tạo để chạy được trang, thật khó để có thể viết các test chỉ tập trung vào một khía cạnh nào đó của ứng dụng. Và vì thế, kiểm thử đối với các ứng dụng dứa trên nền tảng Web Forms sẽ khó khăn hơn nhiều so với khi áp dụng trên ứng dụng MVC. Hơn thế nữa, việc kiểm thử trên nền tảng Web Forms yêu cầu phải sử dụng đến web server. Nền tảng MVC phân tách các thành phần và sử dụng các interface (khái niệm giao diện trong lập trình hướng đối tượng), và nhờ đó có thể kiểm thử các thành phần riêng biệt trong tình trạng phân lập với các yếu tố còn lại của ứng dụng.  
  
Sự phân tách rạch ròi ba thành phần của ứng dụng MVC còn giúp cho việc lập trình diễn ra song song. Ví dụ như một lập trình viên làm việc với view, lập trình viên thứ hai lo cài đặt logic của controller và lập trình viên thứ ba có thể tập trung vào logic tác vụ của model tại cùng một thời điểm.  
  
Lựa chọn áp dụng MVC trong xây dựng ứng dụng  
  
Bạn cần phải xem xét kỹ càng việc áp dụng mô hình ASP.NET MVC hay mô hình ASP.NET Web Forms khi xây dựng một ứng dụng. Mô hình MVC không phải là mô hình thay thế cho Web Forms, bạn có thể dùng một trong hai mô hình.  
  
Trước khi quyết định sử dụng MVC hay Web Forms cho một web site cụ thể, bạn cần phải phân tích lợi ích khi chọn một trong hai hướng.

*Lợi ích của ứng dụng web dựa trên mô hình MVC*  
  
Nền tảng ASP.NET MVC mang lại những lợi ích sau:

* Dễ dàng quản lý sự phức tạp của ứng dụng bằng cách chia ứng dụng thành ba thành phần model, view, controller
* Nó không sử dụng view state hoặc server-based form. Điều này tốt cho những lập trình viên muốn quản lý hết các khía cạnh của một ứng dụng.
* Nó sử dụng mẫu Front Controller, mẫu này giúp quản lý các requests (yêu cầu) chỉ thông qua một Controller. Nhờ đó bạn có thể thiết kế một hạ tầng quản lý định tuyến.
* Hỗ trợ tốt hơn cho mô hình phát triển ứng dụng hướng kiểm thử (TDD)
* Nó hỗ trợ tốt cho các ứng dụng được xây dựng bởi những đội có nhiều lập trình viên và thiết kế mà vẫn quản lý được tính năng của ứng dụng

*Lợi ích của ứng dụng được xây dựng trên nền tảng Web Forms*

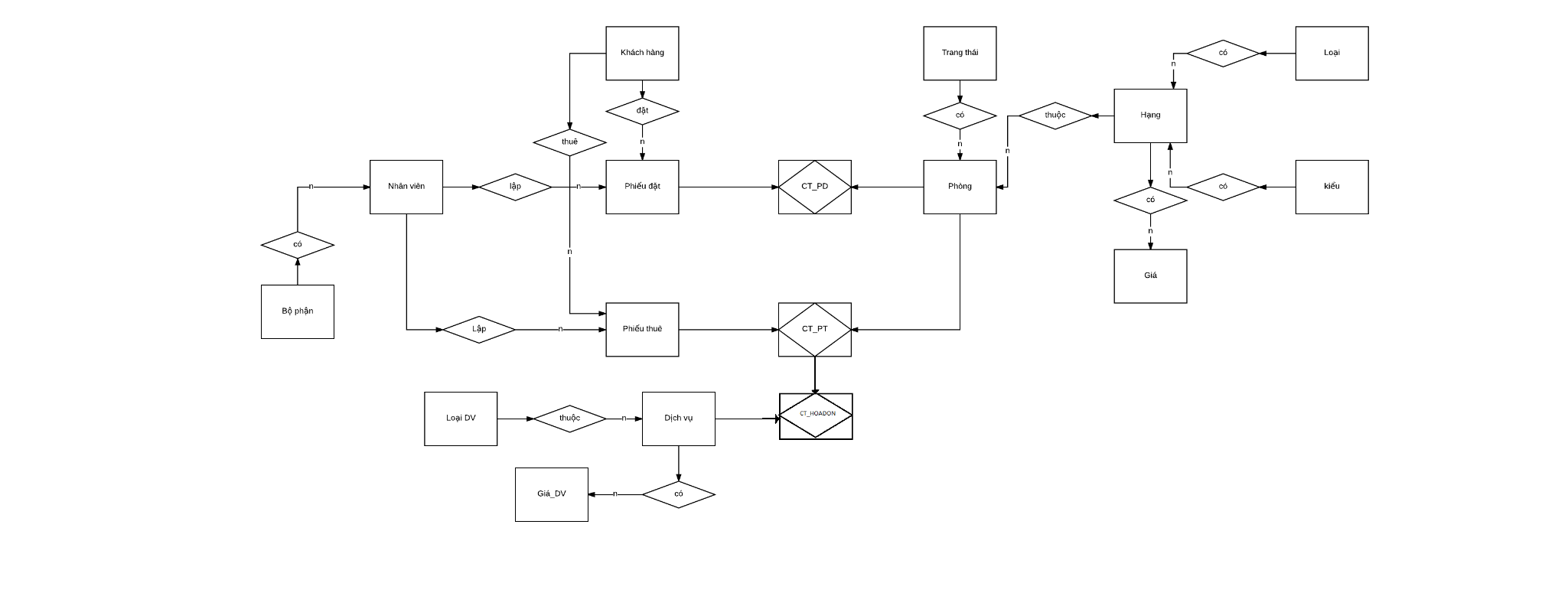
* Nó hỗ trợ cách lập trình hướng sự kiện, quản lý trạng thái trên giao thức HTTP, tiện dụng cho việc phát triển các ứng dụng Web phục vụ kinh doanh. Các ứng dụng trên nền tảng Web Forms cung cấp hàng tá các sự kiện được hỗ trợ bởi hàng trăm các server controls.
* Sử dụng mẫu Page Controller.
* Mô hình này sử dụng view state hoặc server-based form, nhờ đó sẽ giúp cho việc quản lý trạng thái các trang web dễ dàng.
* Nó rất phù hợp với các nhóm lập trình viên quy mô nhỏ và các thiết kế, những người muốn tận dụng các thành phần giúp xây dựng ứng dụng một cách nhanh chóng.
* Áp dụng Web Forms giúp giảm bớt sự phức tạp trong xây dựng ứng dụng, bởi vì các thành phần (lớp Page, controls,…) được tích hợp chặc chẽ và thường thì giúp bạn viết ít code hơn là áp dụng theo mô hình MVC.

Các tính năng của nền tảng ASP.NET MVC

* Tách bạch các tác vụ của ứng dụng (logic nhập liệu, business logic, và logic giao diện), dễ dàng kiểm thử và mặc định áp dụng hướng phát triển TDD. Tất cả các tính năng chính của mô hình MVC được cài đặt dựa trên interface và được kiểm thử bằng cách sử dụng các đối tượng mocks, mock object là các đối tượng mô phỏng các tính năng của những đối tượng thực sự trong ứng dụng. Bạn có thể kiểm thử unit-test cho ứng dụng mà không cần chạy controller trong tiến trình ASP.NET, và điều đó giúp unit test được áp dụng nhanh chóng và tiện dụng. Bạn có thể sử dụng bất kỳ nền tảng unit-testing nào tương thích với nền tảng .NET.
* MVC là một nền tảng khả mở rộng (extensible) & khả nhúng (pluggable). Các thành phần của ASP.NET MVC được thiết kể để chúng có thể được thay thế một cách dễ dàng hoặc dễ dàng tùy chỉnh. Bạn có thể nhúng thêm view engine, cơ chế định tuyến cho URL, cách kết xuất tham số của action-method và các thành phần khác. ASP.NET MVC cũng hỗ trợ việc sử dụng Dependency Injection (DI) và Inversion of Control (IoC). DI cho phép bạn gắn các đối tượng vào một lớp cho lớp đó sử dụng thay vì buộc lớp đó phải tự mình khởi tạo các đối tượng. IoC quy định rằng, nếu một đối tượng yêu cầu một đối tượng khác, đối tượng đầu sẽ lấy đối tượng thứ hai từ một nguồn bên ngoài, ví dụ như từ tập tin cấu hình. Và nhờ vậy, việc sử dụng DI và IoC sẽ giúp kiểm thử dễ dàng hơn.
* ASP.NET MVC có thành phần ánh xạ URL mạnh mẽ cho phép bạn xây dựng những ứng dụng có các địa chỉ URL xúc tích và dễ tìm kiếm. Các địa chỉ URL không cần phải có phần mở rộng của tên tập tin và được thiết kế để hỗ trợ các mẫu định dạng tên phù hợp với việc tối ưu hóa tìm kiếm (URL) và phù hợp với lập địa chỉ theo kiểu REST.
* Hỗ trợ sử dụng đặc tả (các thẻ) của các trang ASP.NET(.aspx), điều khiển người dùng (.ascx) và trang master page (.marter). Bạn có thể sử dụng các tính năng có sẵn của ASP.NET như là sử dụng lồng các trang master page, sử dụng in-line expression (<%= %>), sử dụng server controls, mẫu, data-binding, địa phương hóa (localization) và hơn thế nữa.
* Hỗ trợ các tính năng có sẵn của ASP.NET như cơ chế xác thực người dùng, quản lý thành viên, quyền, output caching và data caching, seession và profile, quản lý tình trạng ứng dụng, hệ thống cấu hình…
* ASP.NET MVC 3 còn bổ sung một view engine mới là Razor View Engine cho phép thiết lập các view nhanh chóng, dễ dàng và tốn ít công sức hơn so với việc sử dụng Web Forms view engine.

1. **MÔ HÌNH CƠ SỞ DỮ LIỆU – QUẢN LÝ KHÁCH SẠN**

ERD Quản lý khách sạn



1. Diagram Quản lý khách sạn

