

LỜI NÓI ĐẦU

Trong bối cảnh phát triển của đất nước, việc tin học hóa các hệ thống quản lý trở nên thiết thực hơn bao giờ hết. Hệ thống quản lý được tin học hóa giúp chúng ta dễ dàng thao tác, mang đến giao diện trực quan sinh động và hấp dẫn. Hơn nữa việc tin học hóa cũng giúp các sản phẩm của chúng ta đến với người dùng một cách rộng rãi hơn, góp phần làm tăng mức độ phổ biến của sản phẩm.

Vì vậy, em quyết định thực hiện một trang web cung cấp các trò chơi Flash, mang lại một sân chơi bổ ích cho những người thường xuyên sử dụng máy tính.

Trang web có hai phần, phần chơi game và phần quản trị game.

Phần chơi game, người dùng cần đăng ký tài khoản, sau đó chọn game để chơi. Việc phân loại game theo nhiều tiêu chí (game ngẫu nhiên, game nổi bật, game mới, game theo thể loại ...) giúp người chơi dễ dàng tìm được game yêu thích của mình.

Phần quản trị game cung cấp giao diện quản lý cho quản trị viên. Quản trị viên được phân cấp thành hai mức, mức quản trị viên nội dung có một số quyền về chỉnh sửa game và thể loại game, mức quản trị trang web có toàn quyền thay đổi tất cả thuộc tính của game, thể loại game, tài khoản người dùng.

Ngoài ra ở phần chơi game, em đã kết hợp với một hệ thống SMS với đầu số 8022 của Vinasoft để người chơi nạp tài khoản.

Trong quá trình thực hiện đồ án tốt nghiệp, em đã nhận được nhiều đánh giá sâu sắc và góp ý thực tế của cô giáo Tiến sĩ Nguyễn Thanh Huyền, bộ môn Toán Tin khoa Toán Tin Ứng dụng, trường Đại học Bách khoa Hà Nội. Em xin gửi lời cảm ơn chân thành tới cô.

PHẦN I. GIỚI THIỆU ỨNG DỤNG WEB VIẾT BẰNG RUBY ON RAILS

I. Ngôn ngữ Ruby

Ruby là 1 ngôn ngữ lập trình hướng đối tượng được tác giả Yukihiro “Matz” Matsumoto bắt đầu phát triển vào năm 1993 và được phổ biến rộng rãi phiên bản đầu tiên năm 1995, tên của ngôn ngữ Ruby là 1 cách chơi chữ với ngôn ngữ đã có từ trước đó là Perl (ngọc). Hiện nay phiên bản mới nhất của Ruby là phiên bản 3.0.

Ruby là 1 ngôn ngữ thông dịch nên có thể mềm dẻo sử dụng trên nhiều hệ điều hành khác nhau (Windows, Linux, Mac OS...)

II. Ruby on Rails

Ruby on Rails là 1 hệ thống cho phép phát triển ứng dụng Web một cách nhanh chóng, Ruby on Rails bao gồm 2 phần:

- Phần ngôn ngữ Ruby.
- Phần framework Rails bao gồm nhiều thư viện.

III. Cấu trúc của ứng dụng Ruby on Rails

1 ứng dụng Web viết bằng Ruby on Rails dựa theo cấu trúc MVC: Model – View – Controller.

Trong đó:

- Phần Model: chứa các lớp phục vụ thao tác trực tiếp với cơ sở dữ liệu
- Phần View: là phần mã HTML (có thể nhúng mã Ruby) để hiển thị nội dung trang web phía trên máy khách

- Phần Controller: chứa các lớp nhận request từ máy khách, sau đó kết nối với các lớp phần Model để lấy dữ liệu từ server và sau đó chuyển dữ liệu tìm được cho phần View.

PHẦN II. KHẢO SÁT HỆ THỐNG

I. Mục đích của đề tài

1. Mục đích

Xây dựng ứng dụng Web là 1 website cung cấp các trò chơi flash, tích hợp thêm nạp tài khoản bằng tin nhắn SMS vào đầu số 8022 và phần quản trị cho quản trị viên.

2. Chức năng chính

Liệt kê danh sách các game flash phục vụ người chơi đăng kí tài khoản để chơi; thêm/trừ số tiền trong tài khoản khi người chơi thắng/thua; cho phép người chơi nạp thêm tiền vào tài khoản bằng cách gửi tin nhắn tới đầu số 8xxx.

II. Đối tượng cần lưu trữ trong CSDL

1. Game

Thông tin về game được lưu trong CSDL bao gồm:

- Tên game (tên ngắn gọn khoảng 20 kí tự)
- media_id (dãy số sinh dựa trên tên game, xác định duy nhất 1 game)
- Mô tả game (đoạn text giới thiệu nội dung game)
- Tên file thực thi game (file flash có phần mở rộng .swf)
- Tên file thumb của game (file ảnh có phần mở rộng .jpg, .png, .gif)
- Số lần được chơi.

- Ấn (true/false, có hiện ra trong danh sách game của người dùng hay không)
- Chi phí mua game.
- Ngày giờ tạo
- Ngày giờ cập nhật

2. Thẻ loại game

- Tên thẻ loại (tên ngắn gọn khoảng 20 kí tự)
- media_id (dãy số sinh dựa trên tên thẻ loại, xác định duy nhất 1 thẻ loại)
- Ấn (true/false, có hiện ra trong danh sách game của người dùng hay không)
- Ngày giờ tạo
- Ngày giờ cập nhật

3. Tài khoản người dùng

- username
- password
- media_id (dãy số sinh dựa trên username, xác định duy nhất 1 user)
- email
- Số điểm hiện tại
- Số tiền trong tài khoản
- Ngày giờ tạo
- Ngày giờ cập nhật

4. Tin nhắn SMS

- Số điện thoại người nhắn
- Số điện thoại dịch vụ
- media_id của user cần nạp tiền
- Nội dung tin nhắn vào
- Nội dung tin nhắn ra

III. Các thao tác nghiệp vụ

1. Trên đối tượng game

- Tìm kiếm
- Chơi
- Cập nhật (tên, mô tả, thể loại, ẩn/hiện)
- Thêm (upload thêm game lên server)
- Xóa (xóa game khỏi server)

2. Thể loại game

- Tạo mới
- Xóa
- Cập nhật (tên, mô tả, ẩn/hiện)

3. Tài khoản người dùng

- Tạo tài khoản mới
- Đăng nhập/Đăng xuất
- Xem thông tin tài khoản của mình

- Thay đổi thông tin cá nhân
- Cộng điểm
- Cộng/Trừ tiền

IV. Những người sử dụng hệ thống

1. Người chơi

- Đăng kí, đăng nhập, đăng xuất
- Chính sửa thông tin cá nhân
- Chọn game (trong menu thả xuống, trong danh sách game ngẫu nhiên hoặc bất kì game nào được liệt kê sẵn trên màn hình)
- Chơi game
- Nạp tiền vào tài khoản

2. Người quản trị nội dung (Mod)

- Chính sửa thông tin về game (tên game, mô tả, thể loại của game, ẩn/hiện game)
- Chính sửa thông tin về thể loại game (tên thể loại, ẩn/hiện thể loại)
- Tạo mới thể loại.

3. Người quản trị trang web (Admin)

Quản trị trang web có các quyền của người quản trị nội dung và có thêm các quyền sau:

- Tạo tài khoản người dùng mới
- Xóa tài khoản người dùng

- Chỉnh sửa thông tin trong tài khoản người dùng
- Thêm game mới
- Xóa game
- Thêm thể loại mới
- Xóa thể loại
- Xem thống kê chi phí mua games, doanh thu do người dùng nhắn tin SMS.

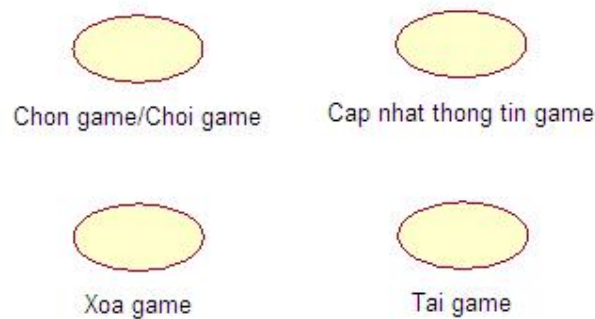
PHẦN III. PHÂN TÍCH HỆ THỐNG

I. Xây dựng biểu đồ Use Case

Danh sách tác nhân: Người chơi (User), người Quản trị nội dung (Mod), người Quản trị Web (Admin)

Danh sách các Use Case (viết tắt là UC): từ bản khảo sát, ta xác định được các chức năng của hệ thống, ta chia chúng thành các gói để dễ quan sát, các gói là: *gói game, gói thể loại game, gói người chơi, gói nhân viên quản trị nội dung, gói nhân viên quản trị, gói nạp tài khoản*.

1. Gói Game



Hình 1. UC của gói game

- UC “Choi game”:
 - Mô tả: Người chơi chọn game bất kì được hiển thị dạng thumbnail hoặc trong danh sách của menu thả xuống
 - Tác nhân kích hoạt: Người chơi
 - Tiền điều kiện: Trang web được load đầy đủ, kết nối tới server thành công.

- Dòng sự kiện:
 - + Người chơi click vào hình đại diện của game hoặc chọn game trong danh sách của menu thả xuống.
 - + Hiện màn hình chơi game phục vụ người chơi.
 - + UC kết thúc.
- UC “Tải game”:
 - Mô tả: Tải game mới lên máy chủ.
 - Tác nhân: Nhân viên quản trị trang Web (Admin)
 - Tiền điều kiện: Kết nối tới server thành công.
 - Dòng sự kiện:
 - + Đăng nhập tài khoản admin.
 - + Kiểm tra có đúng tài khoản admin không.
 - + Hiện thị link dẫn vào khu vực quản trị.
 - + Admin click vào mục “Quản lý game”
 - + Admin click vào nút “Tải game”
 - + Admin nhập các thông tin yêu cầu (tên game, mô tả game, chọn file game .swf, chọn file hình ảnh đại diện cho game .jpg, .bmp, .png)
 - + Admin ấn nút “Tải lên”
 - + Tải game lên server
 - + Hiện thị thông báo thành công/thất bại của việc upload.

+ UC kết thúc.

- UC “Cập nhật thông tin game”:

- Mô tả: Cập nhật, chỉnh sửa các thông tin sai lệch về game.
- Tác nhân: Mod, Admin.
- Tiền điều kiện: đăng nhập thành công với tài khoản mod/admin.
- Dòng sự kiện:

+ Mod/Admin click vào “Quản lý game”

+ Hiển thị danh sách các game trên server để mod/admin lựa chọn

+ Mod/Admin tiến hành chỉnh sửa tên/mô tả cho game

+ Mod/Admin click nút “Cập nhật” tương ứng của game (mỗi game có 1 nút “Cập nhật” riêng nhằm sử dụng công nghệ AJAX)

+ Cập nhật thông tin mới từ các ô dữ liệu trong form lên CSDL máy chủ.

+ Hiển thị thông báo cập nhật thành công.

+ UC kết thúc.

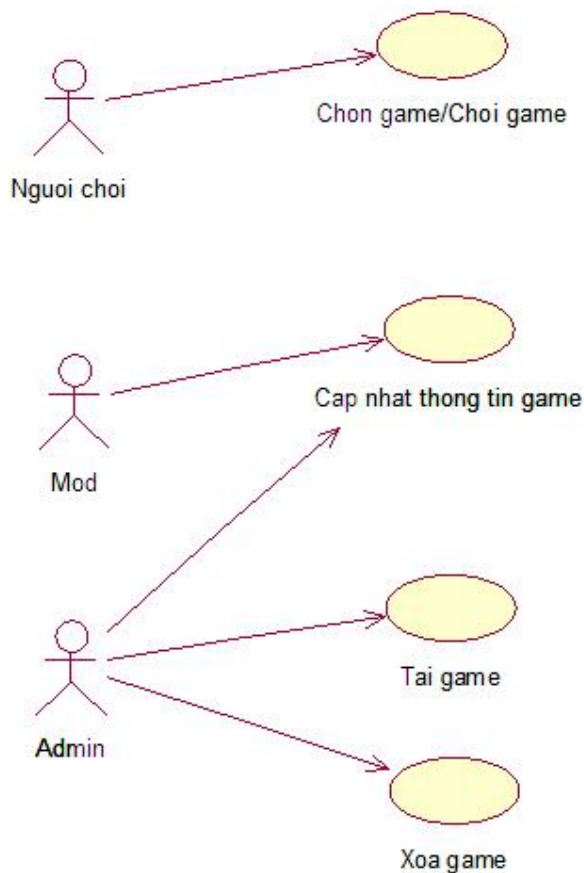
- UC “Xóa game”:

- Mô tả: Admin xóa game khỏi máy chủ.
- Tác nhân: Admin
- Tiền điều kiện: đăng nhập thành công với tài khoản Admin.
- Dòng sự kiện:

+ Admin click “Quản lý game” để vào khu vực quản lý game.

- + Hiển thị danh sách toàn bộ game trên server.
- + Lựa chọn game cần xóa.
- + Click nút “Xóa game”
- + Xóa game khỏi máy chủ (các file .swf, file hình ảnh .jpg, các trường trong CSDL)
- + UC kết thúc.

Ta sơ đồ biểu diễn quan hệ các UC trong gói game:



Hình 2. Quan hệ các UC trong gói game

2. Gói Thẻ loại Game



Hình 3. UC của gói thẻ loại game

- UC “Thêm mới”:
 - Mô tả: Admin/Mod tạo thêm category.
 - Tác nhân: Admin/Mod
 - Tiền điều kiện: đăng nhập thành công với tài khoản Admin/Mod
 - Dòng sự kiện:
 - + Admin/Mod click “Quản lý Thẻ loại game” để vào khu vực quản lý thẻ loại.
 - + Hiện thị danh sách toàn bộ thẻ loại.
 - + Admin/Mod nhập tên thẻ loại game cần tạo vào ô input.
 - + Click nút “Tạo mới”.
 - + Quay trở lại màn hình hiển thị danh sách thẻ loại game (có kèm theo thông báo về kết quả tạo thẻ loại có thành công hay không)
 - + UC kết thúc.

Mặc định thẻ loại vừa tạo ra sẽ có thuộc tính “Phát hành” nhận giá trị false, nghĩa là thẻ loại game này sẽ không hiện ra trên màn hình của người chơi.

- UC “Cập nhật”:

- Mô tả: Admin/Mod cập nhật thông tin về thể loại.
- Tác nhân: Admin/Mod
- Tiền điều kiện: đăng nhập thành công với tài khoản Admin/Mod
- Dòng sự kiện:
 - + Admin/Mod click “Quản lý Thể loại game” để vào khu vực quản lý thể loại.
 - + Hiện thị danh sách toàn bộ thể loại.
 - + Nhập tên mới cho thể loại, hoặc tick vào checkbox “phát hành”
 - + Ấn nút “Cập nhật” để cập nhật các thông tin vừa thay đổi.
 - + UC kết thúc.

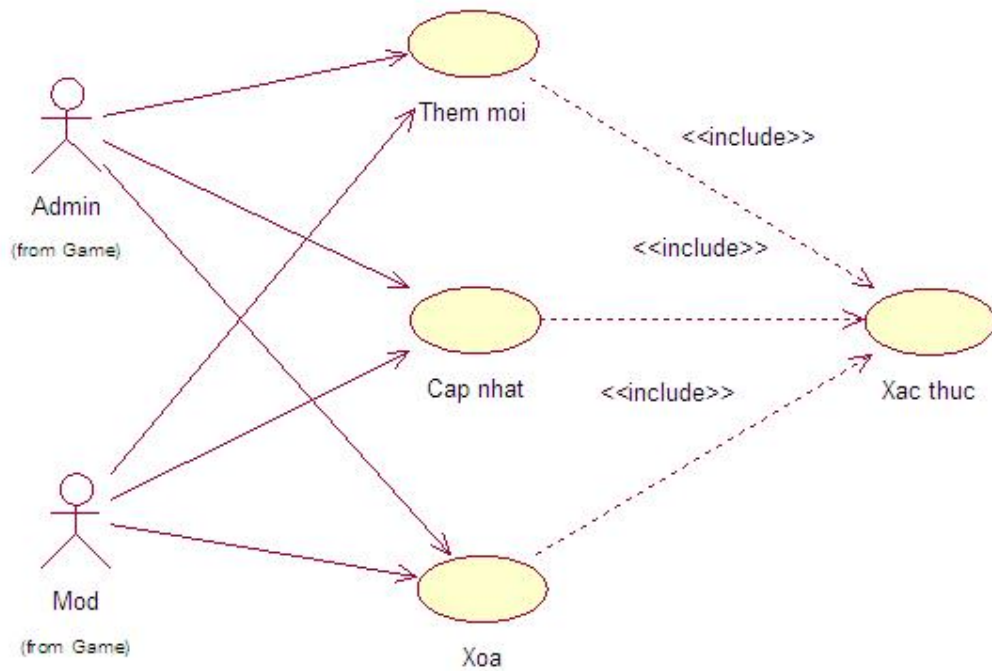
- UC “Xóa”:

- Mô tả: xóa 1 thể loại game.
- Tác nhân: Admin/Mod
- Tiền điều kiện: đăng nhập thành công với tài khoản Admin/Mod
- Dòng sự kiện:
 - + Admin/Mod click “Quản lý Thể loại game” để vào khu vực quản lý thể loại.
 - + Hiện thị danh sách toàn bộ thể loại.
 - + Ấn nút “Xóa” để xóa thể loại game.

+ Hiển thị danh sách mới.

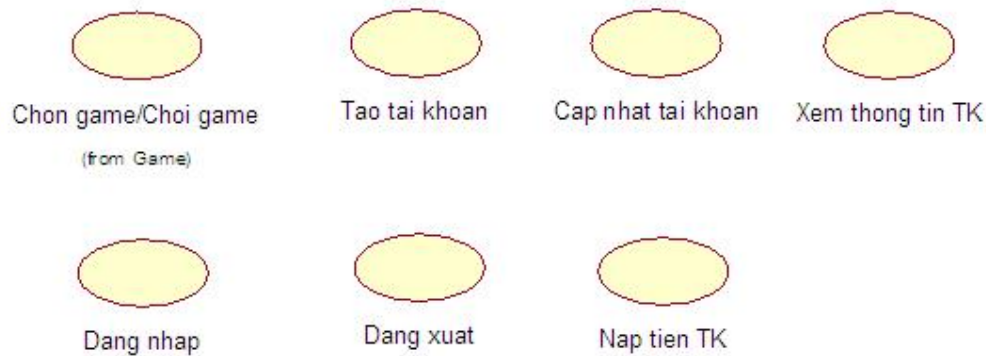
+ UC kết thúc.

Ta có sơ đồ biểu diễn quan hệ các UC trong gói “thẻ loại game”:



Hình 4. Quan hệ của các UC trong gói thẻ loại game

3. Gói Người chơi



Hình 5. UC của gói người chơi

UC Chọn game/Chơi game đã được mô tả trong gói Game, ta mô tả 4 UC còn lại là “Tạo tài khoản”, “Xem thông tin TK”, “Cập nhật TK”, “Đăng nhập”, “Đăng xuất” và “Nạp tiền TK”

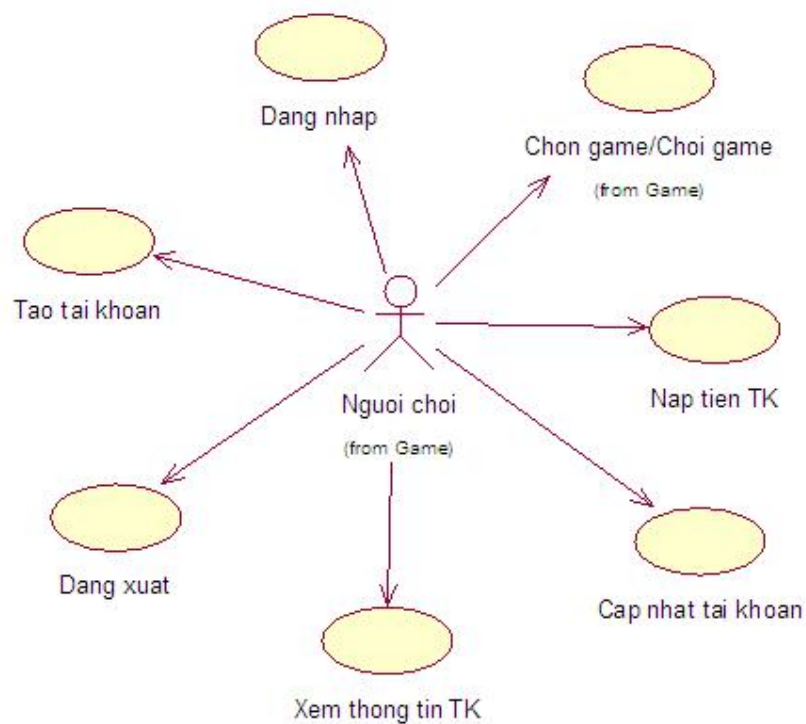
- UC “Tạo tài khoản”:
 - Mô tả: Khách thăm trang web tạo tài khoản cá nhân.
 - Tác nhân: Khách thăm trang web
 - Tiền điều kiện: trang web được tải đầy đủ.
 - Dòng sự kiện:
 - + Khách click nút “Đăng ký”
 - + Hiện form để khách điền thông tin về username, password...
 - + Khách ấn nút “Đăng ký”
 - + Kiểm tra các trường bắt buộc phải nhập, nếu có trường trống thì thông báo yêu cầu nhập đầy đủ.

- + Nếu đã đủ thông tin, gửi thông tin lên server để tạo mới tài khoản
- + Đăng nhập cho người dùng vào tài khoản vừa tạo
- + Hiện trang chủ của site
- + UC kết thúc.
- UC “Xem thông tin tài khoản”:
 - Mô tả: User xem thông tin về tài khoản mình đăng nhập như: username, password, số điểm, các game đã chơi, userID, tài khoản hiện tại
 - Tác nhân: Người chơi.
 - Tiền điều kiện: Đăng nhập thành công với 1 tài khoản.
 - Dòng sự kiện:
 - + Người dùng click vào nút “Xem thông tin”
 - + Hiện thông tin cá nhân
 - + UC kết thúc.
- UC “Cập nhật tài khoản”:
 - Mô tả: người dùng cập nhật thông tin về tài khoản của mình
 - Tác nhân: người dùng
 - Tiền điều kiện: đăng nhập thành công
 - Dòng sự kiện:
 - + click nút “Cập nhật tài khoản”
 - + Hiện ra form thông tin các dữ liệu của tài khoản đó

- + Người dùng chỉnh sửa thông tin
- + Người dùng ấn nút “Cập nhật”
- + thông tin được kiểm tra, cập nhật lên server, nếu có trường trống yêu cầu nhập thêm.
- + Hiện thông báo cập nhật thành công.
- + UC kết thúc
- UC “Đăng nhập”:
 - Mô tả: Khách thăm trang web đăng nhập tài khoản đã được tạo từ trước.
 - Tác nhân: Khách thăm trang web
 - Tiền điều kiện: trang web được tải đầy đủ.
 - Dòng sự kiện:
 - + Khách nhập thông tin về username vào form
 - + Tiếp tục nhập thông tin về password
 - + Ấn phím Enter trên bàn phím để gửi request tới server
 - + Nếu thông tin username/password là chính xác, server phản hồi lại thông tin và đăng nhập người dùng vào hệ thống
 - + Nếu thông tin username/password không chính xác, hiện thông báo đăng nhập không thành công
 - + UC kết thúc.

- UC “Đăng xuất”:
 - Mô tả: Người dùng đang đăng nhập, muốn thoát đăng nhập
 - Tác nhân: Người dùng đang đăng nhập trang web
 - Tiền điều kiện: trang web được tải đầy đủ.
 - Dòng sự kiện:
 - + Người dùng click vào nút “Đăng xuất”
 - + Hệ thống thực hiện xóa các thông tin đăng nhập của người dùng
 - + Cập nhật lại trang web, hiện form đăng nhập
 - + UC kết thúc.
- UC “Nạp tiền TK”:
 - Mô tả: Người chơi nạp tiền vào tài khoản bằng tin nhắn SMS
 - Tác nhân: Người bất kì
 - Tiền điều kiện: tạo thành công 1 tài khoản trên site và có số userID xác định
 - Dòng sự kiện:
 - + Người dùng sử dụng điện thoại di động nhắn tin với cú pháp “VNS NAPTIENTEN XXX” với XXX là userID của tài khoản người dùng
 - + Tin nhắn được hệ thống xử lý tin nhắn xử lý
 - + Nếu đúng cú pháp, không có lỗi, tài khoản người dùng sẽ được tăng thêm 1 khoảng YYY nào đó.
 - + UC kết thúc.

Sơ đồ quan hệ các UC trong gói “Người chơi”:



Hình 6. Quan hệ của các UC trong gói người chơi

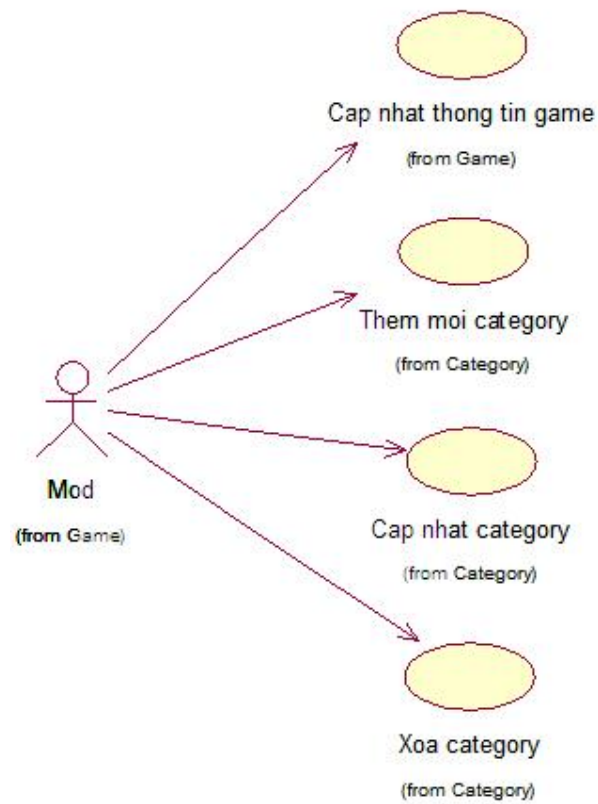
4. Gói nhân viên quản trị Nội dung



Hình 7. UC của gói nhân viên quản trị nội dung

Các UC này đã được mô tả trong gói “Game” và gói “Thể loại game”

Ta có sơ đồ biểu diễn quan hệ của gói “Nhân viên quản trị Nội dung”:



Hình 8. Quan hệ của các UC trong gói nhân viên quản trị nội dung

5. Gói Nhân viên quản trị trang Web



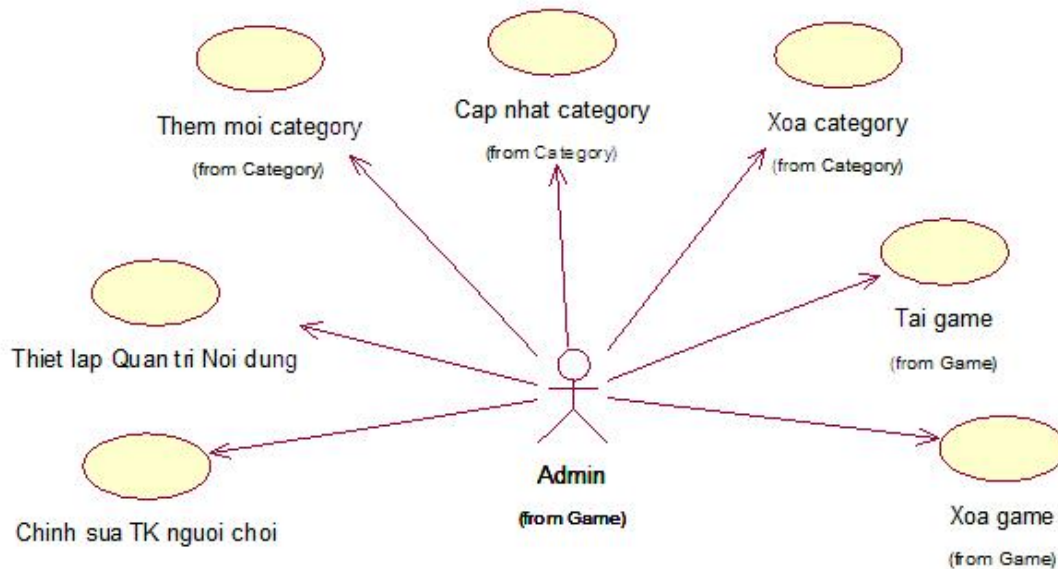
Hình 9. UC của gói nhân viên quản trị Web

Các UC về game và thể loại game đã được mô tả ở phần gói game và gói thể loại game.

- UC “Chỉnh sửa TK người chơi”:
 - Mô tả: Chỉnh sửa, cập nhật, xóa tài khoản người chơi
 - Tác nhân: Admin
 - Tiền điều kiện: Đăng nhập thành công tài khoản Admin
 - Dòng sự kiện:
 - + Click vào mục “Quản lý”
 - + Hiện danh sách các mục cần quản lý
 - + Chọn mục “User”

- + Hiện danh sách các tài khoản đã đăng kí
- + Ấn nút “Edit”/”Delete” để cập nhật, xóa tài khoản người chơi
- + Khi ấn nút “Edit”, hiện bảng thông tin người dùng để Admin chỉnh sửa, sau đó Admin ấn nút “Cập nhật” để xác nhận thông tin mới cần cập nhật.
- + UC kết thúc.
- UC “Thiết lập Quản trị nội dung”:
 - Mô tả: thiết lập cho 1 tài khoản nào đó nhận quyền người Quản trị nội dung.
 - Tác nhân: Admin
 - Tiền điều kiện: Đăng nhập thành công tài khoản Admin
 - Dòng sự kiện:
 - + Click vào mục “Quản lý”
 - + Hiện danh sách các mục cần quản lý
 - + Chọn mục “User”
 - + Hiện danh sách các tài khoản đã đăng kí
 - + Cập nhật phần “Vai trò” thành “Quản trị Nội dung”
 - + Ấn nút “cập nhật” để xác nhận.
 - + UC kết thúc.

Ta có sơ đồ quan hệ các UC trong gói “Quản trị trang web”:



Hình 10. Quan hệ các UC trong gói nhân viên quản trị Web

6. Gói Nạp tài khoản



Hình 11. UC của gói nạp tài khoản

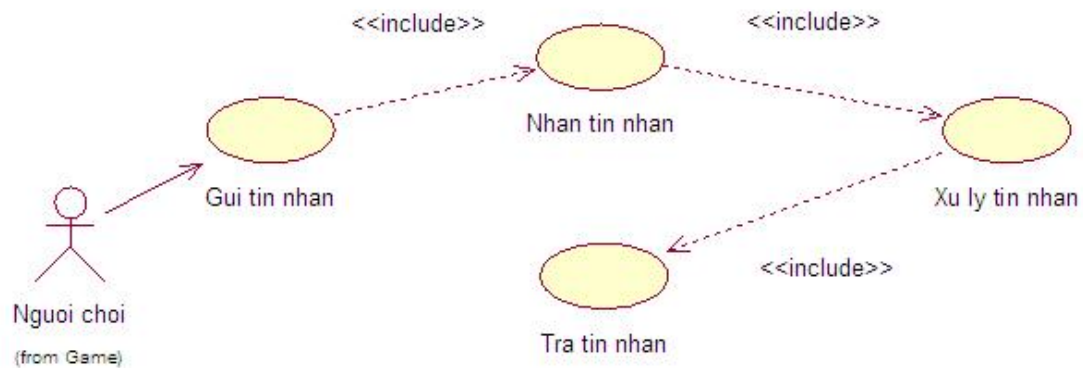
Trong đó 2 UC “Nhận tin nhắn” và “Trả tin nhắn” thuộc hệ thống gateway của 1 nhà cung cấp dịch vụ viễn thông nào đó. Gateway này sẽ được ta thuê của nhà cung cấp đó. 3 UC “Nhận tin nhắn”, “Xử lý tin nhắn”, “Trả tin nhắn” là UC trừu tượng, sẽ được kích hoạt khi người dùng kích hoạt UC “Gửi tin nhắn”

- UC “Gửi tin nhắn”:

- Mô tả: Người dùng gửi tin nhắn theo cú pháp đã được ta qui định trước tới đầu số 8xxx (hoặc 6xxx) để nạp tiền.
- Tác nhân: Người chơi
- Tiền điều kiện: hệ thống gateway của nhà cung cấp hoạt động tốt.
- Dòng sự kiện:
 - + Người dùng sử dụng ĐTDĐ soạn tin theo cú pháp đã qui định trước (ví dụ: NAP <userID>, với userID là giá trị của trường userID trong CSDL, ta sẽ cung cấp cho người dùng giá trị này thông qua màn hình của UC “Xem thông tin tài khoản” trong gói “Khách”) tới số 8xxx (hoặc 6xxx)
 - + UC “Nhận tin nhắn” được kích hoạt, phần này nằm trong hệ thống SMS Gateway của nhà cung cấp dịch vụ viễn thông
 - + UC “Nhận tin nhắn” chuyển đổi từ thông tin SMS sang request HTTP và gửi tới file xử lý tin nhắn SMS của ta
 - + UC “Xử lý tin nhắn” sử dụng file xử lý tin nhắn SMS của ta để phân tích chuỗi kí tự trong tin nhắn, nhận biết userID
 - + UC “Xử lý tin nhắn” kết nối tới CSDL, cập nhật giá trị tài khoản cho userID tương ứng
 - + UC “Xử lý tin nhắn” chuẩn bị nội dung báo cáo “thành công” hoặc “không thành công”
 - + UC “Trả tin nhắn” trả lại người dùng tin nhắn báo cáo.
 - + Người dùng nhận tin nhắn báo cáo.

+ UC kết thúc.

Ta có sơ đồ quan hệ các UC trong gói “Nạp tài khoản”:



Hình 12. Quan hệ của các UC trong gói nạp tài khoản

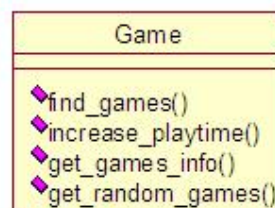
II. Xây dựng biểu đồ lớp

Ta có các gói biểu đồ lớp sau: gói “cơ sở dữ liệu” và gói “tác nghiệp”

1. Gói cơ sở dữ liệu

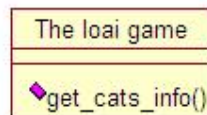
Gói cơ sở dữ liệu bao gồm lớp game, lớp thể loại game, lớp người dùng, lớp thể loại game-game (là lớp thao tác trên bảng kết nối giữa game và thể loại game)

- Lớp game:



Lớp game bao gồm các hàm lấy thông tin về game, thao tác chỉnh sửa các thuộc tính của game trong cơ sở dữ liệu.

- `find_game()`: tìm các game với tiêu chí nào đó. Đầu vào là tham số thể hiện việc cần tìm game mới, game nhiều người chơi hay game ngẫu nhiên, đầu ra là 1 mảng các mảng băm chứa thuộc tính của game.
- `increase_playtime()`: tăng số lần chơi của game. Đầu vào là id của game.
- `get_games_info()`: lấy thông tin về game như id, tên, mô tả, số lần chơi, ảnh. Đầu vào là 1 mảng các đối tượng game, đầu ra là 1 mảng các mảng băm chứa các thuộc tính của các game ở đầu vào.
- `get_random_games()`: lấy ra 1 số game ngẫu nhiên. Đầu vào là số lượng game cần lấy ngẫu nhiên, đầu ra là 1 mảng các mảng băm chứa các thuộc tính của game.
- Lớp “thẻ loại game”:



Lớp thẻ loại game gồm hàm lấy thông tin về thẻ loại game từ cơ sở dữ liệu.

- `Get_cats_info()`: lấy thông tin về thẻ loại game như id, tên, ảnh. Đầu vào là 1 mảng các đối tượng thẻ loại game, đầu ra là 1 mảng các mảng băm chứa các thuộc tính của các thẻ loại game ở đầu vào.

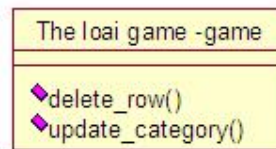
- Lớp “Người dùng”:



Lớp người dùng bao gồm các hàm thao tác trên bản ghi của bảng users như tạo mới tài khoản, cập nhật tài khoản, liệt kê các người chơi có điểm cao nhất, tăng số điểm của người chơi, ...

- create(): tạo mới 1 bản ghi trong bảng users. Đầu vào là các tham số về thông tin của user như: username, password, email.
- update(): cập nhật thông tin của 1 user vào cơ sở dữ liệu. Đầu vào là username, password, email.
- top_players(): lấy ra 10 user có số điểm cao nhất.
- increase_score(): tăng số điểm của 1 user. Đầu vào là id của user đó.
- decrease_credit(): trừ tài khoản của 1 user. Đầu vào là id của user đó.
- authenticate(): kiểm tra xem user có thể login được không. Đầu vào là username, password. Đầu ra trả về 1 đối tượng user tương ứng với username, password đó.
- encrypted_password(): mã hóa password theo SHA1, đầu vào là password, salt, đầu ra là hashed_password (password đã được mã hóa)

- Lớp thể loại game-game:

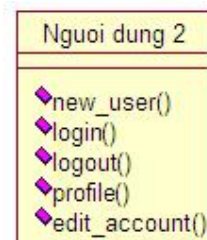


Lớp này gồm các hàm để thao tác trên bảng cats_games tức là bảng join của bảng thể loại games (cats) và bảng games.

- delete_row(): xóa 1 cặp game-thể loại game trong bảng cats_games. Đầu vào là game_id và cat_id.
- update_category(): thêm 1 cặp game-thể loại game vào bảng cats_games. Đầu vào là game_id và cat_id. Đầu ra là giá trị Boolean thể hiện việc thêm thành công hay không.

2. Gói tác nghiệp

- Lớp “người dùng”:



Lớp người dùng gồm các hàm nhận request từ phía client, tức là phía người sử dụng trang web, rồi giao tiếp với gói cơ sở dữ liệu để lấy thông tin.

- new_user(): tạo mới 1 tài khoản cho user. Đầu vào là các tham số username, password, email nhận được từ form do người dùng submit lên.
- login(): thực hiện công việc đăng nhập cho user, đầu vào là username, password do người dùng submit từ form.

- logout(): đăng xuất user.
- profile(): chuẩn bị thông tin cho phần View để hiển thị thông tin tài khoản của user (như tên nick, tài khoản, số điểm...)
- edit_account(): chuẩn bị thông tin cho phần View để thể hiện thông tin tài khoản của user về username, password, email.
- Lớp “trang chủ”: bao gồm các hàm xử lý các request trên trang chủ



Lớp trang chủ bao gồm các hàm

- Lớp chung:



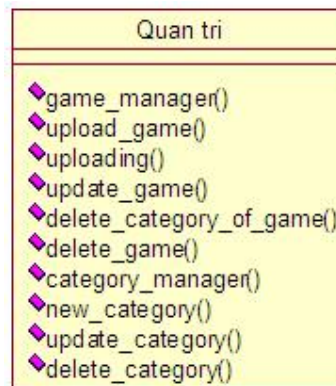
Lớp này bao gồm các hàm dùng chung cho tất cả các lớp

- logged_in?(): kiểm tra xem người dùng có đang đăng nhập hay không.
- playable?(): kiểm tra xem người dùng có được phép chơi game không
- is_admin?(): kiểm tra người dùng đang đăng nhập có phải admin không.

- `is_mod?()`: kiểm tra người dùng đang đăng nhập có phải mod không.
- `require_administrator_roles()`: kiểm tra người dùng đang đăng nhập có phải admin không, nếu không thì đưa người dùng quay lại trang chủ.
- `require_moderator_roles()`: người dùng đang đăng nhập có phải mod không.
- `limit_title()`: cắt ngắn 1 chuỗi ký tự. Đầu vào là 1 chuỗi ký tự, số ký tự cho phép, và phần đuôi muốn thể hiện. Đầu ra là chuỗi sau khi đã cắt ngắn.

- Lớp “quản trị”:

Lớp này chứa các hàm phục vụ công tác quản trị nội dung website như quản lý game, quản lý thể loại game...



- `game_manager()`: lấy thông tin về game để hiển thị cho phần view. Các thông tin cần thể hiện bao gồm tên game, mô tả game, id game, thuộc tính ẩn của game, các thể loại game của game.
- `upload_game()`: hàm này hiển thị form upload game.
- `uploading()`: thực hiện việc tải file lên server.
- `update_game()`: cập nhật thông tin về game. Đầu vào là id game, tên game, mô tả game, thuộc tính ẩn của game, các thể loại của game do quản trị viên submit từ form.

- `delete_category_of_game()`: xóa bỏ 1 thể loại game của game. Đầu vào là id game và id của thể loại game do quản trị viên submit từ form.
- `delete_game()`: xóa game khỏi bảng game. Đầu vào là id của game.
- `category_manager()`: lấy thông tin về thể loại game để hiển thị cho phần view.
- `new_category()`: tạo mới 1 thể loại game. Đầu vào là tên thể loại game.
- `update_category()`: cập nhật thông tin thể loại game. Đầu vào là tên thể loại game, tính ẩn.
- `delete_category()`: xóa 1 thể loại game. Đầu vào là id của thể loại game.

PHẦN IV. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

I. Thiết kế CSDL

1. Các bảng trong cơ sở dữ liệu

Trong đó cột Field (trường), Type (kiểu dữ liệu), Collation (Mã font), Attributes (thuộc tính), Null (có được nhận giá trị Null mặc định hay không), Default (giá trị mặc định), Extra (thông tin thêm)

Bảng games:

Field	Type	Collation	Attributes	Null	Default	Extra
<u>id</u>	int(11)			No	None	auto_increment
title	varchar(255)	utf8_general_ci		Yes	NULL	
media_id	int(11)			Yes	NULL	
description	text	utf8_general_ci		Yes	NULL	
swf	varchar(255)	utf8_general_ci		Yes	NULL	
image	varchar(255)	utf8_general_ci		Yes	NULL	
hidden	tinyint(1)			Yes	1	
price	decimal(4,2)			Yes	NULL	
playtimes	int(11)			Yes	0	
created_at	datetime			Yes	NULL	
updated_at	datetime			Yes	NULL	

Id: id của bản ghi game, có kiểu dữ liệu là số nguyên, không được phép nhận giá trị NULL, không có giá trị mặc định, trường id sẽ được tự động tăng dần mỗi khi có bản ghi mới thêm vào bảng cơ sở dữ liệu, đây là khóa chính của bảng.

Title: tên game, có kiểu dữ liệu là xâu độ dài lớn nhất 255 ký tự, kiểu mã hóa ký tự là utf8_general_ci, có thể nhận giá trị là NULL, nếu quản trị viên không nhập tên cho game thì title sẽ nhận giá trị là 1 xâu rỗng.

Media_id: id của game, được sinh ra từ tên game nhờ hàm **hash** của Ruby, có kiểu dữ liệu là số nguyên, có thể nhận giá trị NULL.

Description: mô tả game, có kiểu dữ liệu là kiểu văn bản, kiểu mã hóa kí tự là utf8_general_ci, có thể nhận giá trị NULL

Swf: tên file .swf quản trị viên upload lên server, kiểu dữ liệu là xâu kí tự độ dài lớn nhất 255, mã hóa kí tự là utf8_general_ci, có thể nhận giá trị NULL.

Image: tên file hình ảnh để hiển thị của game, kiểu dữ liệu là xâu kí tự độ dài tối đa 255, mã hóa kí tự là utf8_general_ci, có thể nhận giá trị NULL.

Hidden: thuộc tính ẩn của game, có kiểu dữ liệu là kiểu Boolean, giá trị mặc định là TRUE (tức là game mới được upload lên server sẽ ẩn đối với người dùng)

Price: chi phí mua game, kiểu dữ liệu số thực có 2 chữ số phần thập phân, giá trị mặc định là NULL.

Playtimes: số lần game được chơi, kiểu dữ liệu số nguyên, giá trị mặc định là 0.

Created_at: thời điểm bản ghi của game được tạo, kiểu dữ liệu là kiểu Datetime, giá trị mặc định là NULL.

Updated_at: thời điểm bản ghi của game được cập nhật, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Bảng users (tài khoản người dùng)

Field	Type	Collation	Attributes	Null	Default	Extra
id	int(11)			No	<i>None</i>	auto_increment
username	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>	
hashed_password	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>	
user_id	int(11)			Yes	<i>NULL</i>	
email	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>	
score	int(11)			Yes	<i>NULL</i>	
credit	int(11)			Yes	<i>NULL</i>	
role_id	int(11)			Yes	<i>NULL</i>	
salt	varchar(255)	utf8_general_ci		Yes	<i>NULL</i>	
created_at	datetime			Yes	<i>NULL</i>	
updated_at	datetime			Yes	<i>NULL</i>	

Id: id của bản ghi, kiểu dữ liệu là kiểu số nguyên, không nhận giá trị NULL, không có giá trị mặc định, tự động tăng thêm 1 đơn vị khi có 1 bản ghi mới được thêm vào bảng, đây là khóa chính của bảng.

Username: username của user, kiểu dữ liệu là kiểu kí tự, mã hóa kí tự theo utf8_general_ci, có thể nhận giá trị NULL.

Hashed_password: mật khẩu của user, kiểu dữ liệu là kiểu kí tự, mã hóa kí tự theo utf8_general_ci, có thể nhận giá trị NULL.

User_id: id của user, kiểu dữ liệu là số nguyên, có thể nhận giá trị NULL.

Email: email của user, kiểu dữ liệu là chuỗi kí tự độ dài lớn nhất 255, mã hóa kí tự theo utf8_general_ci, có thể nhận giá trị NULL.

Score: số điểm của user, kiểu dữ liệu là số nguyên,

Credit: số tiền trong tài khoản của user, kiểu dữ liệu là số nguyên, có thể nhận giá trị NULL.

Role_id: là id trong bảng roles của user, kiểu dữ liệu là kiểu số nguyên, có thể nhận giá trị NULL.

Salt: trường salt hỗ trợ việc tạo password mã hóa, có kiểu dữ liệu là kiểu kí tự, mã hóa kí tự utf8_general_ci, có thể nhận giá trị NULL.

Created_at: thời điểm bản ghi mới được tạo, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Updated_at: thời điểm bản ghi được cập nhật, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Bảng cats (thẻ loại game)

Field	Type	Collation	Attributes	Null	Default	Extra
id	int(11)			No	None	auto_increment
title	varchar(255)	utf8_general_ci		Yes	NULL	
media_id	int(11)			Yes	NULL	
hidden	tinyint(1)			Yes	1	
created_at	datetime			Yes	NULL	
updated_at	datetime			Yes	NULL	

Id: id của bản ghi, kiểu dữ liệu là số nguyên, không nhận giá trị NULL, tự động tăng thêm 1 đơn vị mỗi khi có bản ghi mới được thêm vào bảng, đây là khóa chính của bảng.

Title: tên thẻ loại game, kiểu dữ liệu là kiểu kí tự

Media_id: id của thẻ loại game, kiểu dữ liệu là số nguyên

Hidden: thuộc tính ẩn của thẻ loại game, kiểu dữ liệu Boolean, giá trị mặc định là TRUE

Created_at: thời điểm bản ghi mới được tạo, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Updated_at: thời điểm bản ghi được cập nhật, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Bảng roles (chức vụ)

Field	Type	Collation	Attributes	Null	Default	Extra
<u>id</u>	int(11)			No	None	auto_increment
role_id	int(11)			Yes	NULL	
role_name	varchar(255)	utf8_general_ci		Yes	NULL	
created_at	datetime			Yes	NULL	
updated_at	datetime			Yes	NULL	

Id: id của bản ghi, kiểu dữ liệu là số nguyên, không nhận giá trị NULL, không có giá trị mặc định, tự động tăng 1 đơn vị khi có bản ghi mới thêm vào bảng.

Role_id: id của chức vụ, kiểu dữ liệu là số nguyên.

Role_name: tên của chức vụ, kiểu dữ liệu Boolean, mã hóa kí tự theo mã utf8_general_ci.

Created_at: thời điểm bản ghi mới được tạo, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Updated_at: thời điểm bản ghi được cập nhật, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Bảng cats_games (là bảng join giữa bảng Games và bảng Cats)

Field	Type	Collation	Attributes	Null	Default	Extra
<u>id</u>	int(11)			No	None	auto_increment
game_id	int(11)			Yes	NULL	
cat_id	int(11)			Yes	NULL	

Id: id của 1 cặp game-thẻ loại game (1 bản ghi của bảng `cats_games`), kiểu dữ liệu là số nguyên, không nhận giá trị `NULL`, tự tăng thêm 1 đơn vị khi có bản ghi mới thêm vào bảng.

Game_id: id của bản ghi game, kiểu dữ liệu là số nguyên

Cat_id: id của bản ghi thẻ loại game, kiểu dữ liệu là số nguyên

Bảng `services` (dịch vụ)

Field	Type	Collation	Attributes	Null	Default	Extra
<u>id</u>	int(11)			No	None	auto_increment
user_id	int(11)			Yes	NULL	
phone_number	varchar(255)	utf8_general_ci		Yes	NULL	
service_number	varchar(255)	utf8_general_ci		Yes	NULL	
date	date			Yes	NULL	
content_in	varchar(255)	utf8_general_ci		Yes	NULL	
content_out	varchar(255)	utf8_general_ci		Yes	NULL	
created_at	datetime			Yes	NULL	
updated_at	datetime			Yes	NULL	

Id: id của bản ghi, kiểu dữ liệu là số nguyên, không nhận giá trị `NULL`, không có giá trị mặc định, tự động tăng thêm 1 đơn vị khi có 1 bản ghi mới được thêm vào bảng.

User_id: id của user cần nạp tài khoản, kiểu dữ liệu là kiểu số nguyên.

Phone_number: số điện thoại của người dùng, kiểu dữ liệu là kiểu kí tự.

Service_number: mã dịch vụ, kiểu dữ liệu là kiểu kí tự.

Date: ngày giờ nhận được tin nhắn của người dùng, kiểu dữ liệu là kiểu `Date`.

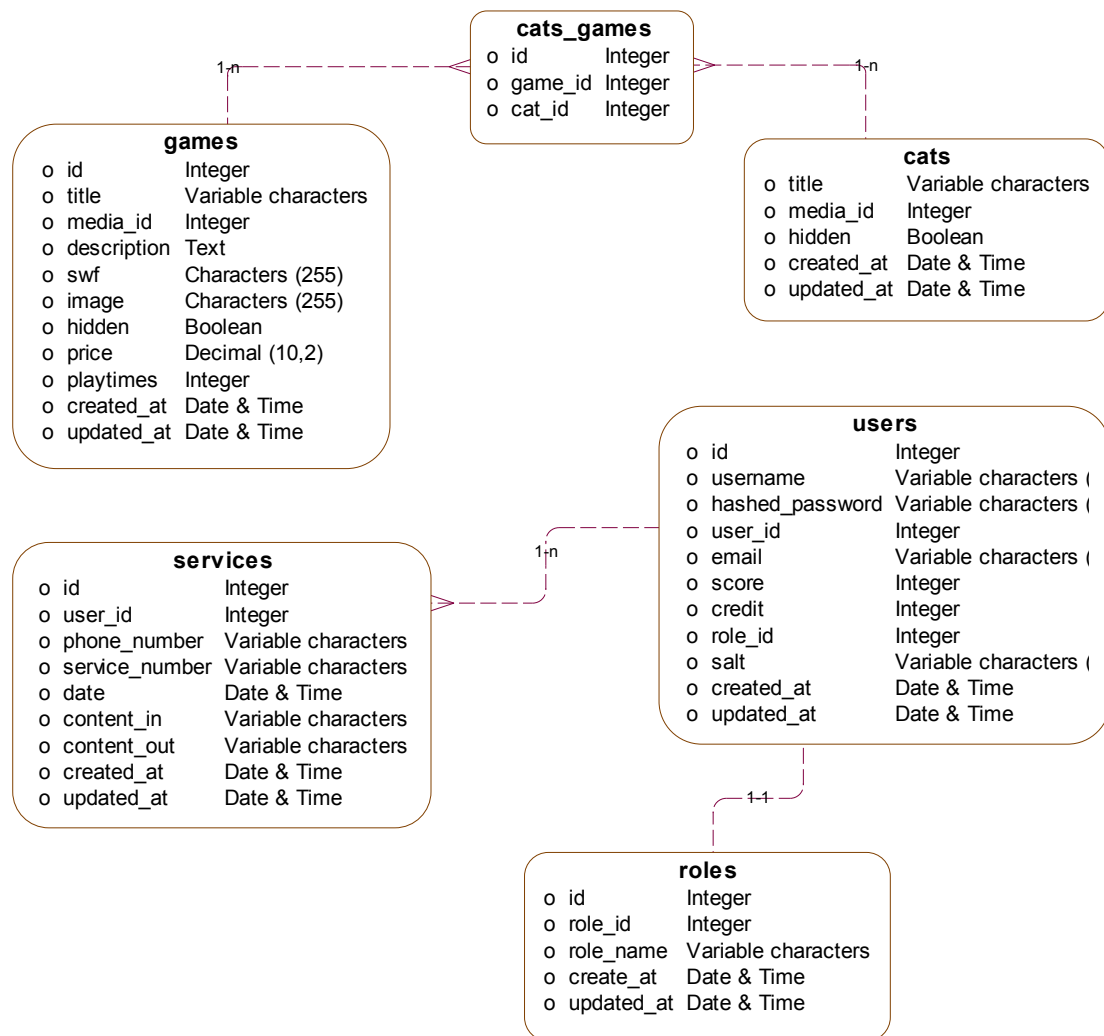
Content_in: nội dung tin nhắn của người dùng nhắn vào hệ thống SMS, kiểu dữ liệu là kiểu kí tự.

Content_out: nội dung tin nhắn trả lại vào điện thoại di động cho người dùng, kiểu dữ liệu là kiểu kí tự.

Created_at: thời điểm bản ghi mới được tạo, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

Updated_at: thời điểm bản ghi được cập nhật, kiểu dữ liệu là Datetime, giá trị mặc định là NULL.

2. Sơ đồ quan hệ giữa các bảng



II. Xây dựng chương trình

Dựa vào biểu đồ lớp ta xây dựng chương trình gồm các gói sau: gói “cơ sở dữ liệu” và gói “tác nghiệp”

1. Gói cơ sở dữ liệu

Gói cơ sở dữ liệu bao gồm lớp game, lớp thể loại game, lớp người dùng, lớp thể loại game-game (là lớp thao tác trên bảng kết nối giữa game và thể loại game)

- Lớp game:

- **find_games()**: đầu vào là các tham số thể hiện thông tin game muốn tìm (chẳng hạn game mới, game có số lần chơi nhiều, game nổi bật..), đầu ra là 1 mảng các game tìm được.

```
def self.find_games(options={})

  type = options[:type] || "new"

  limit = options[:limit] || 5

  case type

  when "new"

    Game.find(:all, :limit => limit, :order => "id DESC",
:conditions => ["hidden = ?", 0])

  when "hot"

    Game.find(:all, :limit => limit, :order => "playtimes DESC", :conditions => ["hidden =
?", 0])

  when "random"

    games, game_list = [], Game.find(:all, :conditions => ["hidden = ?", 0])

    num_game = [game_list.size, limit].min

    while games.size < num_game

      game = game_list[rand(game_list.size)]
```

```
        games << game
    end

    games

end

end
```

Trong hàm này ta nhận giá trị kiểu tìm kiếm dựa vào tham số “type”, phục vụ tìm kiếm game mới, game được nhiều người chơi, hoặc tìm game 1 cách ngẫu nhiên. Nếu gọi hàm không có tham số type thì mặc định tìm kiếm các game mới được tải lên server.

Các game được tìm kiếm là các game có thuộc tính **Ẩn** là nhận giá trị False.

- **increase_playtime()**: tăng giá trị “số lần chơi” của 1 game, đầu vào là id của game cần xử lý.

```
def self.increment_playtimes(id)

    game = Game.find(:first, :conditions => ["id = ?", id])

    game.update_attribute("playtimes", game.playtimes + 1)

    game

end
```

Trong hàm này ta tăng giá trị của game đang được chơi lên 1, dựa vào id của game.

- **get_games_info()**: lấy thông tin của 1 hay nhiều game, đầu vào là 1 hay nhiều đối tượng game, đầu ra là 1 mảng các mảng băm chứa thông tin về các trường của các game đó.

```
def self.get_games_info(games)
```

```
games = [games] unless games.kind_of?(Array)

info = []

games.each do |g|

  info << { :game => g, :id => g.id, :title => g.title, :description => g.description,
:playtimes => g.playtimes, :hidden => g.hidden}

end

info

end
```

Trong hàm này ta có đầu vào là 1 mảng các đối tượng game, sau đó trả lại là 1 mảng các mảng băm bao gồm thông tin về: đối tượng game, id game, tên game, số lần chơi, mô tả game và tính ẩn/hiện của game đó.

- **get_random_games()**: lấy ra 1 số game ngẫu nhiên, đầu ra là 1 mảng các mảng băm chứa thông tin về 5 game được lấy ngẫu nhiên.

```
def self.get_random_games

  rg = Game.find_games(:type => "random", :limit => 5)

  Game.get_games_info(rg)

end
```

Trong hàm này ta lấy ra 5 game ngẫu nhiên, số game giới hạn ở đây là 5 nhằm phục vụ 5 game ngẫu nhiên hiện ra trên trang chủ. Hàm này gọi lại hàm get_games_info để lấy thông tin về 5 game ngẫu nhiên vừa tìm được.

- Lớp “thẻ loại game”:

- **get_cats_info()**: đầu vào là 1 hay nhiều đối tượng “thẻ loại game”, đầu ra là 1 mảng các mảng băm chứa thông tin về thẻ loại game như: đối tượng thẻ loại game, id của thẻ loại, tên thẻ loại, thuộc tính ẩn của thẻ loại.

```
def self.get_cats_info(cats)

  cats = [cats] unless cats.kind_of?(Array)

  info = []

  cats.each { |cat| info << { :cat => cat, :id => cat.id, :title => cat.title, :hidden =>
cat.hidden} }

  info

end
```

- Lớp “Người dùng”:

- **create()**: tạo mới 1 tài khoản người chơi trong cơ sở dữ liệu

```
def self.create(username, password, realname, email, gender, location, aboutme)

  user = User.new(:username => username, :score => 0, :credit => 500, :email => email,
:role_id => 3,:user_id => username.hash.abs)

  user.password = password

  user.hashed_password = encrypted_password(password,user.salt)

  user.save

  user

end
```

Trong hàm này ta tạo ra 1 bản ghi mới cho bảng Users, với thông tin về các trường được cung cấp từ đầu vào của hàm. Hàm này không lưu thẳng password của

người dùng vào bảng Users mà tiến hành mã hóa theo mã SHA rồi mới lưu vào trường password của bảng Users

- **update()**: cập nhật thông tin tài khoản người dùng. Đầu vào là username, password và email của người dùng

```
def self.update(username, password, email)

  user = User.find(:first, :conditions => ["username = ?", username])

  user.hashed_password = encrypted_password(password, user.salt) unless
password.empty?

  user.email = email

  user.save

end
```

Trong hàm này ta cũng tiến hành mã hóa lại password trước khi lưu vào cơ sở dữ liệu, cụ thể là trường password của bảng Users.

- **top_players()**: trả lại 1 mảng chứa danh sách các người chơi có điểm cao nhất, ở đây là 10 người, xếp theo thứ tự điểm từ cao đến thấp

```
def self.top_players

  users = User.find(:all, :limit => 10, :order => "score DESC")

end
```

- **increase_score()**: tăng điểm của người chơi

```
def self.increase_score(userid)

  user = User.find(:first, :conditions => ["user_id = ?", userid])

  user.update_attribute("score", user.score + 1)

end
```

Trong hàm này ta tìm đối tượng user cần tăng điểm dựa trên thông tin là id của user đó, sau đó tiến hành cập nhật trường “score”.

- **decrease_credit()**: trừ tiền trong tài khoản người chơi

```
def self.decrease_credit(userid, minus=1)

  user = User.find(:first, :conditions => ["user_id = ?", userid])

  user.update_attribute("credit", user.credit - 1)

end
```

Cập nhật số tiền trong tài khoản của người chơi, mỗi lần chơi 1 game sẽ bị trừ số tiền đi 1 đơn vị.

- **authenticate()**: đầu vào là cặp username/password, đầu ra là giá trị Boolean thể hiện cặp username/password có tương ứng với 1 tài khoản nào trong cơ sở dữ liệu hay không.

```
def self.authenticate(username, password)

  user = User.find(:first, :conditions => ["username = ?", username])

  if user

    expected_password = encrypted_password(password, user.salt)

    user = nil unless user.hash_password == expected_password

  end

  user

end
```

Trong hàm này ta tìm user có username như tham số đầu vào cung cấp, nếu tìm thấy thì so sánh password của username đó với tham số password, nếu giống

nhau thì trả về đối tượng user đó, nếu không trả về giá trị nil (tương đương FALSE trong ngôn ngữ Ruby)

- Lớp thể loại game-game:

Lớp này gồm các hàm để thao tác trên bảng cats_games tức là bảng join của bảng thể loại games (cats) và bảng games

- **delete_row()**: hàm này thực hiện chức năng xóa 1 hàng trong bảng cats_games, mỗi hàng trong bảng cats_games là 1 bản ghi có trường cat_id và game_id, trong đó cat_id là id của thể loại game, game_id là id của game.

```
def self.delete_row(game_id, cat_id)

  self.find(:first, :conditions => ["game_id = ? and cat_id = ?", game_id, cat_id]).destroy

end
```

- **update_category()**:

```
def self.update_category(game_id, cat_id)

  row = self.find(:first, :conditions => ["game_id = ? and cat_id = ?", game_id, cat_id])

  if row.blank?

    self.create(:game_id => game_id, :cat_id => cat_id)

    return true

  end

  return false

end
```

Hàm này tạo mới 1 bản ghi trong bảng cats_games, dựa trên thông tin về cat_id và game_id được cung cấp trong dãy tham số khi gọi hàm.

2. Gói tác nghiệp

Gói tác nghiệp gồm lớp người dùng, lớp trang chủ, lớp chung.

- Lớp “người dùng”:
 - **new_user()**: nhận các tham số được gửi lên từ form nhập liệu của người dùng, đó là các thông tin về username, password, email

```
def new_user

  if request.post?

    @username = params[:username]

    @password = params[:password1]

    @email = params[:email]

    @errors = {}

    reged_user = User.find(:first, :conditions => ["username = ?", @username])

    @errors[:username] = "<< Mời bạn điền Tên đăng nhập" if (@username.strip == "")

    if (reged_user && @username.strip == reged_user.username)

      @errors[:username] = "<< Tên đăng nhập đã bị sử dụng"

    end

    @errors[:password] = "<< Mời bạn điền Mật khẩu" if @password.strip == ""

    @errors[:email] = "<< Email không hợp lệ" if @email && @email !~ /^[a-zA-Z]+[a-zA-Z0-9+._-]*@[a-zA-Z0-9+._-]+\.[a-zA-Z]{2,5}$/

    @errors[:email] = "<< Mời bạn điền Email" if @email.strip == ""

    if @errors.size.zero?

      user = User.create(@username, @password, @realname, @email, @gender,
        @location, @aboutme)
```



```
        session[:username], session[:user_id], session[:role_id] = @username, user.user_id,
        user.role_id

        redirect_to home_path

    end

end

end
```

Trước tiên hàm này kiểm tra xem request có phải là submit form hay không, nếu đúng như vậy thì tiến hành nhận các giá trị của params username, password, email.

Trong quá trình đăng ký 1 account mới có kèm theo việc kiểm tra sự phù hợp của các thông tin, chẳng hạn như kiểm tra xem tên đăng nhập có hợp lệ không (đảm bảo không để trống, hoặc chưa có ai đăng ký tên này), email có hợp lệ không (email được kiểm tra bằng Regular Expression)

Nếu không có lỗi nào thì tạo mới 1 user bằng cách gọi hàm **create()** của lớp User trong gói “cơ sở dữ liệu”, sau đó tự động đăng nhập bằng tài khoản vừa tạo.

- **login()**: cung cấp form cho người dùng đăng nhập

```
def login

  if request.post?

    username = params[:username].strip

    password = params[:password].strip

    user = User.authenticate(username, password)

    errors = {}

    if user
```

```
session[:username], session[:user_id], session[:role_id] = username, user.user_id,
user.role_id

render :update do |page|

  page.visual_effect :opacity, "user-actions", :from => 1.0, :to => 0.5, :duration => 0.2

  page.replace_html "user-actions", :partial => "user_home_logged_in"

  page.visual_effect :opacity, "user-actions", :from => 0.5, :to => 1.0, :duration => 0.2

end

else

  errors[:username] = username

  render :update do |page|

    page.replace_html "user-actions", :partial => "user_home_not_logged_in",

      :locals => {:errors => errors}

    page.visual_effect :shake, "cabin", :duration => 0.1

    #page << "new Effect.Shake('cabin', {duration:0.1})"

  end

end

end

end

end
```

Hàm này nhận thông tin từ form mà người dùng điền vào và ấn nút Enter để đăng nhập. Sau khi nhận các tham số username, password thì tiến hành tìm trong cơ sở dữ liệu user có username trùng với username nhận được. Nếu đúng password thì đăng nhập cho người dùng, nếu sai sẽ sử dụng AJAX để tạo hiệu ứng rung, đồng thời hiện thông báo “đăng nhập không thành công” lên màn hình.

- **log_out()**: đăng xuất khỏi trang web.

```
def logout

  session[:username] = session[:user_id] = session[:role_id] = nil

  render :update do |page|

    page.visual_effect :opacity, "user-actions", :from => 1.0, :to => 0.5, :duration => 0.2

    page.replace_html "user-actions", :partial => "user_home_not_logged_in", :locals =>
{:errors => ""}

    page.visual_effect :opacity, "user-actions", :from => 0.5, :to => 1.0, :duration => 0.2

  end

end
```

Hàm này xóa thông tin về session đăng nhập của người dùng hiện tại, hiển thị lại cửa sổ đăng nhập.

- **profile()**: cho phép người dùng xem thông tin về tài khoản của mình

```
def profile

  @user = User.find(:first, :conditions => ["user_id = ?", params[:user_id]])

end
```

Hàm này tìm thông tin về user, để cung cấp cho phần thể hiện lên trang web các thông tin của user (như tên, tài khoản hiện tại, nhóm người dùng)

- **edit_account()**: cho phép người dùng sửa thông tin của tài khoản

```
def account

  @user = User.find(:first, :conditions => ["username = ?", session[:username]])

  unless @user.nil?
```

```
unless request.post?

  @password = @user.hashed_password

  @email = @user.email

else

  @password = params[:password1]

  @email = params[:email]

  @errors = {}

  #@errors[:password] = "<< Mời bạn điền Mật khẩu" if (@password &&
  @password.strip == "")

  @errors[:email] = "<< Email không hợp lệ" if @email && @email !~ /^[a-zA-Z]+[a-
  zA-Z0-9+._-]*@[a-zA-Z0-9+._-]+\.[a-zA-Z.]{2,5}$/

  @errors[:email] = "<< Mời bạn điền Email" if (@email && @email.strip == "")

  if @errors.size.zero?

    User.update(@user.username, @password, @email)

  end

end

end

end
```

Hàm này lấy các thông tin về user, hiển thị lên form cho người dùng xem. Người dùng có thể xem hoặc chỉnh sửa rồi cập nhật thông tin mới. Khi người dùng chỉnh sửa thông tin cá nhân và submit thì hàm này sẽ nhận thông tin từ form, tiến hành kiểm tra tính hợp lệ của các thông tin về password, email và cập nhật vào cơ sở dữ liệu.

- Lớp “trang chủ”: bao gồm các hàm xử lý các request trên trang chủ
- **index()**: hàm này gọi các hàm trong gói “cơ sở dữ liệu” để lấy thông tin về các game mới, game ngẫu nhiên, tất cả các game có thuộc tính hiện, các thông tin về thể loại game.

```
def index
```

```
  @random_games = Game.get_random_games
```

```
  hg = Game.find_games(:type => "hot", :limit => 10)
```

```
  @hot_games = Game.get_games_info(hg)
```

```
  @categories = Cat.get_cats_info(Cat.find(:all, :conditions => ["hidden = ?", 0]))
```

```
  @all_games = Game.get_games_info(Game.find(:all, :conditions => ["hidden = ?", 0]))
```

```
end
```

- **refresh_random_games()**:

```
def refresh_random_games
```

```
  games = Game.get_random_games
```

```
  render :update do |page|
```

```
    page.replace_html "list-games", :partial => "game_pix", :locals => {:games => games,  
:delta => 135, :size => 110x110", :crop => "1:1", :name => "thumb110110"}
```

```
    page.visual_effect :opacity, "list-games", :from => 0.1, :to => 1.0, :duration => 0.5
```

```
  end
```

```
end
```

Hàm này lấy các game ngẫu nhiên để thay thế danh sách game ngẫu nhiên đang hiển thị trên màn hình, hiển thị danh sách mới bằng kỹ thuật AJAX.

- **play()**: hiện cửa sổ chơi game

```
def play

  redirect_to_index unless playable?

  game_id = params[:game_id]

  @game = Game.increment_playtimes(game_id)

  @user = User.increment_score(session[:user_id])

  User.credit_minus(session[:user_id], 1)

  game_info = Game.get_games_info(@game)[0]

  html = render_to_string :partial => "play", :locals => {:game_info => game_info}

  render :update do |page|

    page.replace_html "player", html

    page.visual_effect :opacity, "player", :from => 0.2, :to => 1.0, :duration => 0.2

  end

end
```

Hàm này hiện cửa sổ chơi game, kích hoạt file .swf để người dùng chơi game, tăng điểm của người chơi và tăng số lần chơi của game, giảm số tiền trong tài khoản của người chơi.

- **prepare_play()**:

```
def prepare_play

  game_id = params[:game_id]

  if logged_in?

    redirect_to :controller => "home", :action => "play", :game_id => game_id
```

```
else

  render :update do |page|

    page << "alert('#{REQUIRE_LOGIN}')"

  end

end

end
```

Hàm này kiểm tra xem khi ấn vào 1 game để chơi thì người dùng đã Đăng nhập hay chưa, nếu đăng nhập rồi sẽ gọi hàm **play()**, nếu chưa sẽ hiện thông báo yêu cầu đăng nhập.

- **find_random_game():**

```
def find_random_game

  cat_id = params[:cat_id]

  games_in_cat = Cat.find(cat_id).games

  g = games_in_cat[rand(games_in_cat.size)]

  game_id = Game.find(:first, :conditions => ["media_id = ?", g.media_id]).id unless
  g.nil?

  if (logged_in? && !g.nil?)

    redirect_to :controller => "home", :action => "play", :game_id => game_id

  else

    if g.nil?

      render :update do |page|

        page << "alert('#{NOGAME_IN_CAT}')"

      end

    end

  end

end
```

```
end

else

  render :update do |page|

    page << "alert('#{REQUIRE_LOGIN}')"

  end

end

end

end

end
```

Hàm này tìm 1 game ngẫu nhiên nằm trong thể loại game được chọn. Hàm này cũng kiểm tra xem người dùng đã Đăng nhập hay chưa. Nếu người dùng đã Đăng nhập rồi thì hiển thị cửa sổ chơi game.

- **switch_page():**

```
def switch_page

  cat_id = params[:cat_id]

  current_page = params[:current_page].to_i

  max_page = params[:max_page]

  current_page = params[:direction] == "next" ? current_page + 1 : current_page - 1

  games = []

  Cat.find(cat_id).games.each {|g|

    games << Game.get_games_info(Game.find(:first, :conditions => ["gameId = ?",
    g.gameId]))[0]

  }

end
```



```
html = render_to_string(:partial => "cat_content", :locals => {:cat_id  
=> cat_id,:games => games,:current_page => current_page,:max_page => max_page})  
  
render :update do |page|  
  
  page.replace_html "box-center-#{cat_id}", html  
  
  page.visual_effect :opacity, "box-center-#{cat_id}", :from => 0.1, :to => 1.0, :duration  
=> 0.5  
  
end  
  
end
```

Hàm này được gọi khi người dùng ấn vào nút chuyển trang trên cửa sổ của thẻ loại game, kết quả là các game trong trang sau/trước sẽ được hiển thị.

- Lớp chung:

Lớp này bao gồm các hàm dùng chung cho tất cả các lớp.

- **logged_in?()**: hàm này kiểm tra xem có giá trị session[:username] hay không nhằm xác định người dùng đã đăng nhập hay chưa

```
def logged_in?  
  
  return false if session[:username].nil?  
  
  return true  
  
end
```

- **playable?()**:

```
def playable?  
  
  user = User.find(:first, :conditions => ["username = ?", session[:username]])  
  
  return false if user.nil?
```

```
return false if (user.credit.nil? || user.credit < 50)

return true

end
```

Hàm này kiểm tra xem người chơi có quyền chơi hay không, nếu số tiền trong tài khoản nhỏ hơn 50 thì không có quyền, cần phải nạp thêm tiền vào tài khoản thì mới chơi được.

- **is_admin():**

```
def is_admin?

return true if (session[:role_id] && (session[:role_id] == 1))

return false

end
```

Hàm này trả về giá trị boolean xác định xem tài khoản người dùng đang đăng nhập có phải là tài khoản của quản trị trang web không.

- **is_mod():**

```
def is_mod?

logger.info "checking mod => role_id = #{session[:role_id]}"

return true if (session[:role_id] && (session[:role_id] < 3))

return false

end
```

Hàm này trả về giá trị boolean xác định xem tài khoản người dùng đang đăng nhập có phải là tài khoản của quản trị nội dung không.

- **require_administrator_roles():**

```
def require_administrator_roles  
  
  redirect_to_index unless is_admin?  
  
end
```

Hàm này được gọi khi 1 thao tác nào đó yêu cầu quyền admin (quyền của người quản trị trang web)

- **require_moderator_roles():**

```
def require_moderator_roles  
  
  redirect_to_index unless is_mod?  
  
end
```

Hàm này được gọi khi 1 thao tác nào đó yêu cầu quyền mod (quyền của người quản trị nội dung)

- **limit_title():**

```
def limit_title(str, max=14, tail="..")  
  
  str.length > max ? str[0...max-tail.length].strip + tail : str  
  
  return "public/game/gameXml/#{id}/#{id}.xml"  
  
end
```

Hàm này nhận vào 1 chuỗi kí tự (có thể là tên của thể loại game, hoặc tên game), và 1 chuỗi kí tự khác để gắn vào chuỗi kí tự sau khi đã được cắt bỏ, và 1 giá trị max là độ dài lớn nhất của chuỗi kí tự.

• Lớp “quản trị”:

Lớp này chứa các hàm phục vụ công tác quản trị nội dung website như quản lý game, quản lý thể loại game...

- **game_manager():**

```
def flash_manager  
  
  ng = Game.find(:all)  
  
  @new_games = Game.get_games_info(ng)  
  
end
```

Hàm này lấy thông tin về tất cả các game để hiển thị cho người quản trị xử lý.

- **upload_game():**

Bản thân hàm này không chứa nội dung logic, phần hiển thị của hàm này được trình bày trong phần View của cấu trúc MVC.

- **uploading():**

```
def uploading  
  
  unless request.post?  
  
    redirect_to :action => "upload_flash"  
  
    return  
  
  end  
  
  begin  
  
    title = params[:gameName].capitalize || "-"  
  
    desc = params[:gameDesc].capitalize || "-"  
  
    @game = Game.create(params[:game])  
  
    @game.media_id = @game.swf.hash.abs  
  
    @game.title = title
```

```
@game.description = desc

@game.save

flash[:notice] = "Tải thành công."

redirect_to :action => "upload_flash"

rescue Exception => e

  logger.info "error => #{e}"

  flash[:error] = "Có lỗi. Đề nghị xem log!"

  redirect_to :action => "upload_flash"

end

end
```

Hàm này nhận thông tin được submit từ form của hàm **upload_game()**, tạo bản ghi mới cho game và lưu thông tin game vào bảng Games của cơ sở dữ liệu, đồng thời tải game lên server.

- **update_game()**: thay đổi thông tin về game (tên game, mô tả, ẩn/hiện...)

```
def update_game

  id = params[:game_id]

  title = params[:title]

  desc = params[:game_desc]

  cat_list = params[:cat_list]

  game = Game.find(id)

  game.title = title
```

```
game.description = desc

game.hidden = params[:hidden]

game.save

html = ""

unless cat_list.blank?

  for cat_id in cat_list

    if CatsGame.update_category(id, cat_id)

      cat_info = Cat.get_cats_info(Cat.find(cat_id))[0]

      html += "<div id=\"#{id}_belongs_to_cat_#{cat_info[:id]}\">"

      html += "#{cat_info[:title]}"

      html += "<a class=\"link\" onclick=\"new "

      html += "Ajax.Request('/delete_category_of_game/#{id}/#{cat_info[:id]}',"

      html += "{asynchronous:true, evalScripts:true}); return false;\" "

      href="#\">[X]</a>"

      html += "</div>"

    end

  end

end

render :update do |page|

  page.insert_html :after, "selected-cat-#{id}", html

  page << "new Effect.Appear('notice-#{id}');"
```

```
page << "$('notice-#{id}').innerHTML = \"Success\""  
  
page << "new Effect.Fade('notice-#{id}', {duration: 2})"  
  
end  
  
end
```

Hàm này nhận thông tin submit từ form của người dùng, sau đó cập nhật các thông tin được lưu vào cơ sở dữ liệu, rồi hiển thị thông tin mới cập nhật đó lên màn hình.

- **delete_category_of_game()**: xóa game khỏi 1 category nào đó

```
def delete_category_of_game  
  
  game_id = params[:game_id]  
  
  cat_id = params[:cat_id]  
  
  CatsGame.delete_row(game_id, cat_id)  
  
  render :update do |page|  
  
    page.visual_effect :fade, "#{game_id}_belongs_to_cat_#{cat_id}", :duration => 0.2  
  
  end  
  
end
```

Mỗi game có thể thuộc 1 hay 1 vài thể loại game, hàm này sẽ tiến hành loại bỏ tính phụ thuộc của 1 game đối với 1 thể loại game, thông tin nhận vào là các tham số về id của game, id của thể loại game qua form mà quản trị viên submit lên.

- **delete_game()**: xóa 1 game khỏi server

```
def delete_game  
  
  id = params[:game_id]
```

```
game = Game.find(id)
```

```
game.destroy
```

```
render :update do |page|
```

```
  page << "new Effect.Fade('pix-element-#{id}', {duration: 1})"
```

```
end
```

```
end
```

- **category_manager():**

```
def category_manager
```

```
  @categories = Cat.get_cats_info(Cat.find(:all))
```

```
end
```

Lấy tất cả các thể loại game để gửi cho phần View.

- **new_category():** tạo mới thể loại game

```
def new_category
```

```
  cat = Cat.create(:media_id => params[:category].hash.abs)
```

```
  cat.title = params[:category]
```

```
  cat.save
```

```
  @categories = Cat.get_cats_info(Cat.find(:all))
```

```
  render :update do |page|
```

```
    page.replace_html "category-content", :partial => "main_category",
```

```
    :locals => {:categories => @categories}
```

```
    page << "new Effect.Highlight('cat-#{cat.id}')"
```



```
end
```

```
end
```

Hàm này nhận thông tin submit từ form của người quản trị nội dung, sau đó tạo 1 bản ghi mới trong bảng Cats cho thể loại game.

- **update_category()**: cập nhật thông tin (về tên) thể loại game

```
def update_category

  id = params[:cat_id]

  title = params[:cat_title]

  cat = Cat.find(id)

  cat.title = title

  cat.hidden = params[:hidden]

  cat.save

  render :update do |page|

    page << "new Effect.Highlight('cat-#{id}')"

  end

end
```

```
end
```

Hàm này cập nhật thông tin về thể loại game, chẳng hạn như tên thể loại game, tính ẩn/hiện. Các thông tin này được submit từ form.

- **delete_category()**: xóa 1 thể loại game.

```
def delete_category

  id = params[:cat_id]

  cat = Cat.find(id)
```

```
cat.destroy

render :update do |page|

  page << "new Effect.Fade('cat-#{id}', {duration: 0.2})"

end

end
```

Hàm này nhận thông tin submit từ form của quản trị viên, nhận vào tham số cat_id và xóa bản ghi tương ứng trong bảng Cats, sau đó hiển thị danh sách thẻ loại game sau khi đã xóa thẻ loại đó.

III. Xây dựng giao diện

1. Màn hình chính

Màn hình chính gồm có:

- Khu vực “người dùng”:

Form đăng nhập để người dùng nhập thông tin username/passwrod

A screenshot of a web form titled "Cá nhân" (Personal) with a red header. Below the header, there is a link "Đăng ký | Quên Mật khẩu" (Register | Forgot Password). The main section is labeled "Vào tên đăng nhập:" (Enter login name:) and contains a single-line text input field.

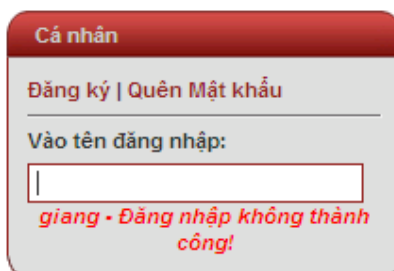
Mô tả hoạt động: Người dùng nhập username vào ô nhập liệu, sau đó ấn Enter, form đăng nhập sẽ yêu cầu nhập password:

A screenshot of the same web form, but now the label "Vào mật khẩu:" (Enter password:) is displayed above the text input field. The "Đăng ký | Quên Mật khẩu" link remains visible above the input field.

Nếu nhập mật khẩu chính xác thì người dùng đăng nhập thành công vào trang Web.

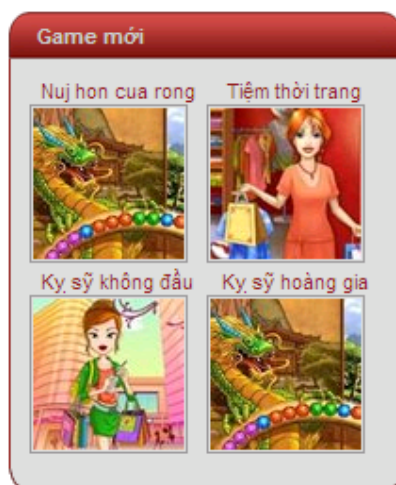


Ngược lại, sẽ có thông báo đăng nhập không thành công:

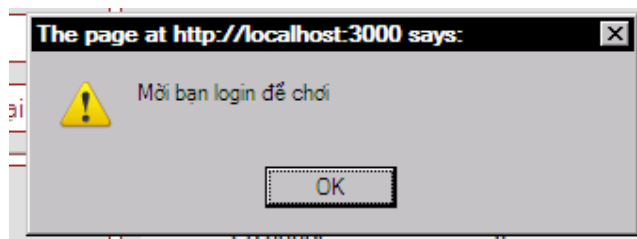


- Khu vực “Game mới”:

Hiển thị danh sách 4 game mới nhất được tải lên server



Khi người dùng đăng nhập và click vào game nào thì sẽ chơi game đó, nếu chưa đăng nhập sẽ hiện thông báo yêu cầu đăng nhập:



- Khu vực “Game ngẫu nhiên”:

Hiển thị 5 game ngẫu nhiên. Có nút ẩn để người dùng click vào sẽ hiển thị 5 game ngẫu nhiên khác



Khi người dùng click vào nút mũi tên (không cần phải đăng nhập) sẽ hiển thị 5 game ngẫu nhiên khác.

- Khu vực “Game được nhiều người chơi”:

Hiển thị 10 game được nhiều người chơi nhất và số lần game đó được chơi

Game HOT nhất		
Game Hot	Lần chơi	Cao thủ
Đào vàng	7	-
Game 5	2	-
Nhà tạo mẫu	1	-
Cuộc chiến không gian	1	-
Cơ nguoi	0	-
Kim cương	0	-
Kỵ sỹ hoàng gia	0	-
Kỵ sỹ không đầu	0	-
Tiệm thời trang	0	-
Nụj hon của rong	0	-

Trong bảng này chia làm 3 cột gồm các thông tin: tên trò chơi, số lần được chơi, tên người chơi có số điểm cao nhất.

- Khu vực “Chơi nhanh”:

Hiển thị danh sách tất cả các game có trên server dưới dạng menu drop-down để người dùng tùy chọn

Hiển thị danh sách tất cả các thể loại game dưới dạng menu drop-down để người dùng tùy chọn, khi chọn 1 thể loại game sẽ tự động chọn 1 game ngẫu nhiên để chuyển người dùng sang màn hình chơi game.

Chơi nhanh

Chọn 1 game

Chọn 1 thể loại

- Khu vực “Thể loại game”:

Hiển thị danh sách tất cả các thể loại game có thuộc tính “Ẩn” là FALSE, mỗi thể loại game hiển thị 4 game, có nút “Xem tất cả” để chuyển người dùng sang màn hình “danh sách game” của thể loại game đó, có 2 nút mũi tên “tiền” và “lùi” để hiện các game trong thể loại game đó theo trang.



2. Màn hình “Game mới”

Hiển thị tất cả các game theo thứ tự từ mới nhất đến cũ nhất

3. Màn hình “Tất cả các game”

Hiển thị tất cả các game theo thứ tự ABC của tên game

4. Màn hình “Thể loại game”

Hiển thị tất cả các game trong 1 thể loại

5. Màn hình “Quản trị”


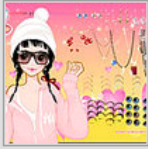
- Màn hình quản trị game:

+ Hiển thị danh sách tất cả các game theo thứ tự tải lên server.

+ Có nút “Upload Game” để quản trị viên chuyển sang màn hình upload game

- + Mỗi game hiện trong 1 khung gồm là 1 form để Quản trị viên cập nhật thông tin về game (tên, mô tả, thuộc thể loại nào, ẩn/hiện) và nút Update, Delete để thực hiện thao tác quản trị.

The screenshot displays a web interface titled "Game management" with a red header bar. Below the header is a button labeled "Upload Game". The main area contains two side-by-side forms for editing game information. Each form includes a game thumbnail, a "Tên:" (Name) field, a "Mô tả:" (Description) field, a "Thể loại:" (Category) dropdown menu, and radio buttons for "Ẩn:" (Hidden) and "Hiện:" (Visible). At the bottom of each form are "Update" and "Delete" buttons.

Game Thumbnail	Tên (Name)	Mô tả (Description)	Thể loại (Category)	Ẩn (Hidden) / Hiện (Visible)	Update Delete
	Đào vàng	Đào vàng kiểu Úc	Game thể thao Game trí tuệ asdfsdf	<input type="radio"/> Ẩn <input checked="" type="radio"/> Hiện	Update Delete
	Nhà tạo mẫu		Game thể thao Game trí tuệ asdfsdf	<input type="radio"/> Ẩn <input checked="" type="radio"/> Hiện	Update Delete

- Màn hình “tải game”:
 - + Hiện 1 form yêu cầu quản trị viên nhập “tên game”, “mô tả game”, đường dẫn tới file thực thi game (.swf), đường dẫn tới file ảnh. Nút “Tải game” và nút “Hủy bỏ”

New game ...

Tên game:

Mô tả ngắn gọn:

File game (.SWF):

Hình ảnh (.JPG, .PNG, .BMP, .GIF):

Back Upload

- Màn hình “Quản lý thể loại game”:

+ hiện danh sách các thể loại game, mỗi thể loại hiển thị trong 1 ô bao gồm “tên thể loại”, thuộc tính ẩn/hiện, nút “cập nhật” và “Xóa” để quản trị viên thực hiện thao tác cập nhật, xóa

+ hiển thị 1 ô nhập liệu để quản trị viên nhập tên của thể loại mới muốn tạo; nút “Tạo mới” để tạo thể loại game có tên như trong ô nhập liệu.

ID	Category properties	Actions
1	Game thể thao Ẩn <input type="radio"/> <input checked="" type="radio"/> Hiện	Delete Update
3	Game trí tuệ Ẩn <input type="radio"/> <input checked="" type="radio"/> Hiện	Delete Update
4	asdfsd Ẩn <input type="radio"/> <input checked="" type="radio"/> Hiện	Delete Update
-	<input type="text"/> Ẩn <input checked="" type="radio"/> <input type="radio"/> Hiện	Add new

- Màn hình “Quản lý doanh thu”:
 - + hiển thị menu tháng, năm để quản trị viên chọn (chẳng hạn 01/2010, 02/2010...)
 - + sau khi quản trị viên chọn 1 tháng, sẽ hiện danh sách gồm chi phí mua game, doanh thu từ tin nhắn SMS người dùng nhấn vào gateway.

KẾT LUẬN

Trong khuôn khổ đồ án tốt nghiệp, em đã tiến hành giới thiệu về ngôn ngữ Ruby và framework Rails phục vụ việc lập trình website. Trang web thực hiện việc cung cấp game cho người chơi phía client, đồng thời cung cấp chức năng quản trị cho các quản trị viên. Một ý nghĩa có thể áp dụng trong thực tế đó là khả năng phát triển về mặt kinh tế của website, khi các trò chơi trở nên hấp dẫn sẽ thu hút được nhiều người chơi, kéo theo số lượng tin nhắn SMS để nạp tài khoản tăng cao.

Tuy nhiên kinh nghiệm còn hạn chế nên hệ thống còn một số điểm chưa được hoàn thiện, vì vậy em sẽ cố gắng tìm hiểu những điều kiện thực tế để có thể áp dụng và nâng cao hệ thống, vươn tới một hệ thống mang có khả năng thống kê chi phí game và số tin nhắn của người chơi một cách chính xác, hiệu quả, đồng thời thu hút được nhiều người tham gia trang web.

Em xin chân thành cảm ơn Cô giáo Tiến sĩ Nguyễn Thanh Huyền đã đọc bản thảo báo cáo và đóng góp chỉ bảo nhiều ý kiến quý báu cho em trong thời gian hoàn thành Đồ án.

PHỤ LỤC

I. Các hình minh họa chương trình

1. Toàn bộ website



2. Phần quản trị

ADMIN

Game

Phân Quyền

Chi Phí Game

Thể Loại Game

Tài Khoản Người Dùng

Tin nhắn nạp tiền

Filter:

Users

Id	User	Username	Role	Score	Credit	Email	Created At	Updated At	
1	375689648	giang	1	12	488	giang@giang.com	Fri May 21 02:17:30 +0700 2010	Sun May 23 17:41:44 +0700 2010	 



3. Phần chơi game



II. Các plugins

Chương trình có sử dụng 1 số plugins của framework Rails như:

1. File_column

Hỗ trợ việc upload file lên server.

2. Streamlined

Hỗ trợ việc tạo bảng quản trị viên đối với các bảng trong cơ sở dữ liệu.

III. Tính chi phí – lợi nhuận từ tin nhắn của người dùng

Tính lợi nhuận bằng cách lấy doanh thu do các tin nhắn người dùng gửi SMS tới đầu số 8xxx trừ đi chi phí mua game.

Đối với các tin nhắn SMS gửi tới đầu số 8xxx, người dùng sẽ mất 1 khoản phí nhất định chẳng hạn 81xx phí là 1000 VND, 82xx là 2000 VND, 83xx là 3000 VND.... số tiền này chia làm 2 phần, nhà cung cấp dịch vụ viễn thông 1 phần và ta thu lại 1 phần

TÀI LIỆU THAM KHẢO

- Đặng Văn Dũng, *Giáo trình Phân tích thiết kế hướng đối tượng bằng UML*, Nhà xuất bản Giáo Dục, 2008.
- Nguyễn Phước Anh Vũ, Phương Lan, *Lập trình ứng dụng Web với RUBY ON RAILS*, Nhà xuất bản Lao động – Xã hội, 2008.
- Website: <http://api.rubyonrails.org>
- Website: <http://railsforge.com>
- Website: <http://rubylearning.org>

MỤC LỤC

LỜI NÓI ĐẦU	1
PHẦN I. GIỚI THIỆU ỨNG DỤNG WEB VIẾT BẰNG RUBY ON RAILS	2
I. Ngôn ngữ Ruby	2
II. Ruby on Rails	2
III. Cấu trúc của ứng dụng Ruby on Rails.....	2
PHẦN II. KHẢO SÁT HỆ THỐNG.....	4
I. Mục đích của đề tài.....	4
1. Mục đích	4
2. Chức năng chính.....	4
II. Đối tượng cần lưu trữ trong CSDL.....	4
1. Game	4
2. Thể loại game	5
3. Tài khoản người dùng	5
4. Tin nhắn SMS	6
III. Các thao tác nghiệp vụ.....	6
1. Trên đối tượng game	6
2. Thể loại game	6
3. Tài khoản người dùng	6
IV. Những người sử dụng hệ thống..Text.....	7
1. Người chơi.....	7
2. Người quản trị nội dung (Mod)	7
3. Người quản trị trang web (Admin).....	7
PHẦN III. PHÂN TÍCH HỆ THỐNG	9

I.	Xây dựng biểu đồ Use Case.....	9
1.	Gói Game.....	9
2.	Gói Thẻ loại Game	13
3.	Gói Người chơi	16
4.	Gói nhân viên quản trị Nội dung	20
5.	Gói Nhân viên quản trị trang Web	22
6.	Gói Nạp tài khoản	24
II.	Xây dựng biểu đồ lớp	26
1.	Gói cơ sở dữ liệu	26
2.	Gói tác nghiệp.....	29
PHẦN IV. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG.....		33
I.	Thiết kế CSDL	33
1.	Các bảng trong cơ sở dữ liệu.....	33
2.	Sơ đồ quan hệ giữa các bảng	39
II.	Xây dựng chương trình	40
1.	Gói cơ sở dữ liệu	41
2.	Gói tác nghiệp.....	48
III.	Xây dựng giao diện	66
1.	Màn hình chính	66
2.	Màn hình “Game mới”	70
3.	Màn hình “Tất cả các game”.....	70
4.	Màn hình “Thẻ loại game”	70
5.	Màn hình “Quản trị”	70
KẾT LUẬN		74
PHỤ LỤC		75
I.	Các hình minh họa chương trình	75

1. Toàn bộ website.....	75
2. Phần quản trị.....	77
3. Phần chơi game.....	77
II. Các plugins.....	78
1. File_column.....	78
2. Streamlined.....	78
III. Tính chi phí – lợi nhuận từ tin nhắn của người dùng.....	78
TÀI LIỆU THAM KHẢO	79
MỤC LỤC.....	80