# HỌC PHẦN THAY THẾ TỐT NGHIỆP 2 CHUYÊN NGÀNH CÔNG NGHỆ PHẦN MỀM

PHÂN TÍCH THIẾT KẾ

ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

**NGUYỄN MẠNH HÙNG ĐỖ THỊ BÍCH NGỌC**

# GIỚI THIỆU

**TÀI LIỆU THAM KHẢO**

1. Object-Oriented and Classical Software Engineering, Stephen R. Schach, Eigtth Edition, Mc Graw Hill, 2010.
2. . *Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers*, Murali Chemuturi, J. Ross Publication Inc., 2011.

MỤC LỤC

[HỌC PHẦN THAY THẾ TỐT NGHIỆP 2 CHUYÊN NGÀNH CÔNG NGHỆ PHẦN MỀM 1](#_Toc87752694)

[GIỚI THIỆU 1](#_Toc87752695)

[MỤC LỤC 2](#_Toc87752696)

[CHƯƠNG 5: CÀI ĐẶT HỆ THỐNG 3](#_Toc87752697)

[5.1 TỔ CHỨC DỰ ÁN 3](#_Toc87752698)

[5.2 CÀI ĐẶT CÁC MODULE 4](#_Toc87752699)

[5.2.1 Cấu hình và kết nối đến cơ sở dữ liệu 4](#_Toc87752700)

[5.2.2 Chức Thống kê tổng số thẻ phạt của các đội bóng 7](#_Toc87752701)

[Lớp TranDauDoiBongDAO.java 7](#_Toc87752702)

[Giao diện thống kê tổng số thẻ phạt 7](#_Toc87752703)

[5.2.3 Chức năng Xem chi tiết thẻ phạt của từng đội 9](#_Toc87752704)

[5.3 KIỂM THỬ ĐƠN VỊ 11](#_Toc87752705)

[5.3.1 Xây dựng bộ test case cho kiểm thử đơn vị 12](#_Toc87752706)

[5.3.2 Cài đặt kiểm thử đơn vị 12](#_Toc87752707)

[5.4 CÂU HỎI ÔN TẬP 14](#_Toc87752708)

[BÀI TẬP DỰ ÁN 15](#_Toc87752709)

# CHƯƠNG 3: THU THẬP VÀ PHÂN TÍCH YÊU CẦU

## THU THẬP YÊU CẦU

## Tìm hiểu lĩnh vực chuyên môn

Tóm tắt các bước thực hiện:

* + - * Bước 1: Làm việc nhóm (có thể dùng brainstorming) để liệt kê tất cả các từ khóa chuyên môn trong lĩnh vực của ứng dụng
      * Bước 2: Phân các từ tìm ra thành nhóm cho dễ theo dõi
      * Bước 3: Giải thích nghĩa của từng từ khóa và lập bảng kết quả các từ khóa (glosary list).

Áp dụng các bước trên vào bài toán quản lí đăng kí học tín chỉ của một trường đại học, gộp bước 1 và 2, có thể gom các từ khóa trong lĩnh vực quản lí đăng kí tín chỉ thành ba nhóm như sau:

|  |  |  |
| --- | --- | --- |
| **Con người** | **Hoạt động của con người** | **Vật, đối tượng** |
| Đội bóng  Cầu thủ  Trọng tài  Ban tổ chức | Quản lí trận đấu  Xem thống kê thẻ phạt các đội bóng  Xem bảng xếp hạng các cầu thủ ghi nhiều bàn thắng  Xem bảng xếp hạng đội bóng  Thanh toán tiền bán bản quyền  Kí hợp đồng bản quyền truyền hình  Thanh toán tiền công cho trọng tài  Cập nhật kết quả thi đấu  Phân công tổ trọng tài cho trận đấu  Đăng kí danh sách cầu thủ cho trận đấu  Lên lịch thi đấu | Thẻ  Hóa đơn  Sân đấu |

**1.2 Bảng giải thích ngữ nghĩa chi tiết cho từng khái niệm trong lĩnh vực của ứng dụng:**

|  |  |  |
| --- | --- | --- |
| **Tên Tiếng Việt** | **Tiếng anh** | **Giải thích** |
| Vòng đấu | Round | Giải được chia làm các vòng đấu để có lịch đấu cụ thể |
| Cầu thủ | Player | Người tham gia chính trong các trận đấu |
| Ban tổ chức | Organizer | Những người tổ chức và điều hành hoạt động của giải đấu |
| Hiệu số | Points | Mỗi trận đấu sẽ được ghi lại kết quả thắng thua hòa và sẽ ghi ra số điểm nhất định. Tiện cho việc sắp xếp các đội bóng |
| Thẻ phạt | Card | Được sử dụng trong mỗi trận đấu khi có cầu thủ phạm lỗi |
| Trọng tài | Referee | Người điều khiển trận đấu, ra quyết định trong trận đấu, |
| Hóa đơn | Bill | Hóa đơn là chứng từ do người bán lập, ghi nhận thông tin chi tiền, thu tiền |

## Mô tả hệ thống bằng ngôn ngữ tự nhiên

***Bước 1: Mục đích của hệ thống:*** Hệ thống trang web phục vụ công tác quản lí

kết quả thi đấu của giải ngoại hạng

Module thực hiện: Thống kê thẻ phạt các đội

***Bước 2: Phạm vi hệ thống***: Những người được vào hệ thống và chức năng mỗi người được thực hiện khi vào hệ thống này được quy định như sau:

* + - * Thành viên hệ thống:
        + Đăng nhập
        + Đăng xuất
      * BTC (Ban tổ chức):
        + Đăng nhập
        + Thống kê thẻ phạt các đội
        + Quản lí các đội bóng, lịch thi đấu
        + Xem bảng xếp hạng các đội bóng
        + Cập nhật kết quả thi đấu
        + Lên hóa đơn thanh toán tiền bản quyền, tiền công cho trọng tài
        + Phân công tổ trọng tài cho trận đấu
        + Lên lịch thi đấu

Những chức năng không đề cập đến thì mặc định là không thuộc phạm vi của hệ thống.

***Bước 3: Hoạt động nghiệp vụ của các chức năng***:

Thống kê thẻ phạt các đội:

* + - * BTC chọn menu thống kê → chọn thống kê thẻ phạt theo đội → danh sách các đội bị thẻ phạt hiện ra, mỗi dòng cho 1 đội: Mã, tên đội, tổng số thẻ đỏ , vàng, tổng số lỗi bị phạt. Sắp xếp theo chiều giảm dần tổng số thẻ đỏ, nếu số thẻ đỏ bằng nhau thì sắp xếp theo chiều giảm dần của tổng số thẻ vàng, nếu số thẻ vàng bằng nhau thì sắp xếp theo chiều giảm dần của tổng số lỗi. Đội xếp cuối bảng sẽ được giải FairPlay → BTC click vào 1 dòng của 1 đội → danh sách các trận đấu mà đội có thẻ hiện lên, mỗi trận trên một dòng : id trận, tên đội đối thủ, số lượng thẻ đỏ, số lượng thẻ vàng

### Bước 4: Thông tin các đối tượng cần xử lí, quản lí:

Nhóm các thông tin liên quan đến con người:

* Ban tổ chức: mã, số điện thoại, email.
* Cầu thủ: mã, tên , năm sinh, vị trí, chân thuận, chiều cao, cân nặng, đặc điểm
* Đội bóng: mã đội, tên, trang phục truyền thống, địa chỉ, mô tả
* Trọng tài: vị trí (biên , chính, bàn)

Nhóm các thông tin liên quan đến cơ sở vật chất:

* Sân đấu: mã sân, tên sân, địa chỉ

Nhóm các thông tin liên quan đến đơn vị, tổ chức:

* Hóa đơn: mã hóa đơn, tổng tiền , mục đích

Nhóm các thông tin liên quan đến thống kê:

* Thống kê thẻ phạt theo đội

### Bước 5: Quan hệ giữa các đối tượng, thông tin:

* + - * Một đội bóng có thể có nhiều cầu thủ
      * Một cầu thủ thuộc về 1 đội bóng
      * Một cầu thủ có thể bị phạt nhiều lỗi
      * Một trận đấu có 4 trọng tài
      * Tại 1 thời điểm 1 trọng tài chỉ có thể tham gia vào 1 trận đấu
      * Một giải đấu có nhiều vòng đấu

## Mô tả hệ thống bằng ngôn ngữ UML - use case

Tóm tắt các bước thực hiện để vẽ biểu đồ use case tổng quan:

* + - * Bước 1: Đề xuất các actor. Với mỗi người dùng xác định ở bước 2 của mục 3.1.2, đề xuất thành một actor tương ứng. Nếu các actor có đặc điểm gì chung, có thể đề xuất actor trừu tượng thành actor cha của các actor tương ứng. Ngoài ra, cần xem xét cần có các actor gián tiếp tác động vào để thực hiện các chức năng hay không.
      * Bước 2: Đề xuất use case. Với mỗi chức năng xác định của mục 3.1.2, đề xuất thành một use case tương ứng
      * Bước 3: Mịn hóa các use case. Nếu có ít nhất 2 use case trùng nhau, cần xem xét gộp lại thành 1. Nếu gộp lại gây hiểu nhầm về số các actor tác động vào, thì có thể dùng use case trừu tượng cho các use case giống nhau, mỗi use case con liên quan đến nhóm các actor tương ứng mà thôi.

Tóm tắt các bước thực hiện để vẽ biểu đồ use case chi tiết:

* + - * Bước 1: Trích phần use case của chức năng tương ứng từ biểu đồ use case tổng quan.
      * Bước 2: Phân rã use case chính thành các use case con: mỗi giao diện (hoặc một số giao diện) tương tác với người dùng có thể đề xuất thành một use case con.
      * Bước 3: Xác định quan hệ của use case con với use case chính: generalization, include, hay extend.
      * Bước 4: Gộp các use case con tương tự nhau bằng cách dùng các use case trừu tượng tổng quát hơn.

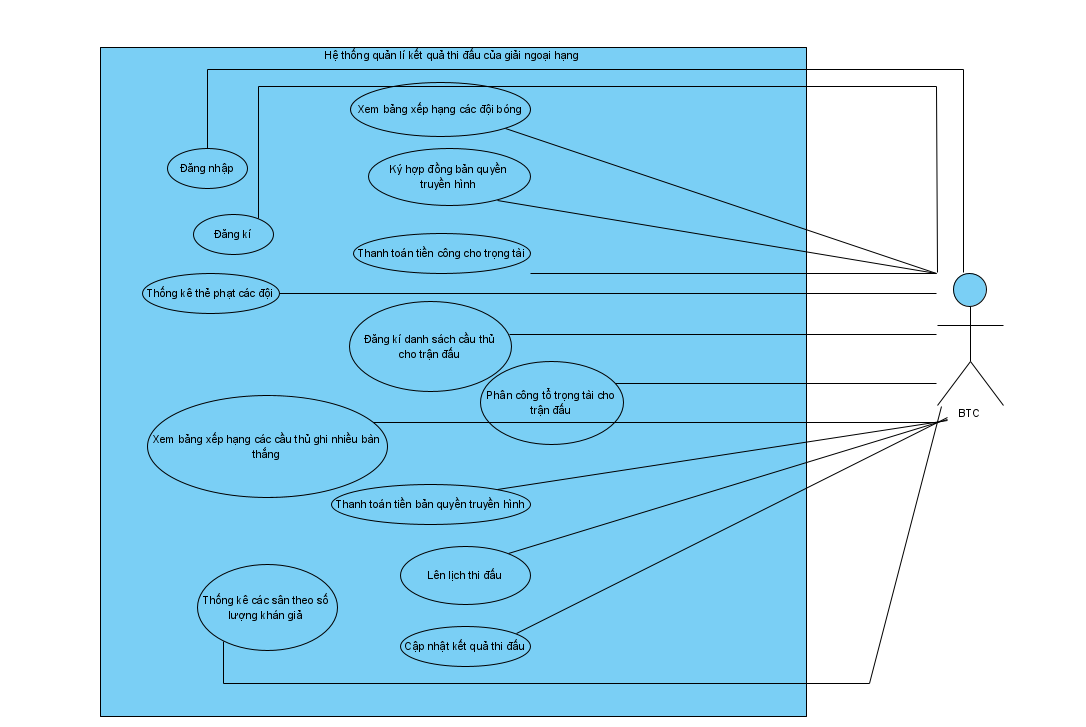
Lưu ý tránh sai sót trong biểu đồ use case:

* + - * Tên use case phải là động từ chỉ hành động của actor. Không nên là động từ chỉ hành động của hệ thống. Cũng không nên là danh từ, tính từ...
      * Mỗi use case phải có tương tác với ít nhất một actor, có thể là trực tiếp hoặc gián tiếp: phải tồn tại ít nhất một đường đi từ một actor nào đó đến use case theo hướng: quan hệ include thì theo chiều mũi tên, quan hệ extend thì ngược chiều mũi tên, quan hệ kế thừa thì gộp lại.

### Biểu đồ use case tổng quan

Ta có thể đề xuất actor của hệ thống : BTC

* + Các chức năng của actor nhân viên: đăng nhập, đăng kí, đổi mật khẩu, Quản lí trận đấu,Xem thống kê thẻ phạt các đội bóng,Xem bảng xếp hạng các cầu thủ ghi nhiều bàn thắng, Xem bảng xếp hạng đội bóng, Thanh toán tiền bán bản quyền, Kí hợp đồng bản quyền truyền hình, Thanh toán tiền công cho trọng tài, Cập nhật kết quả thi đấu, Phân công tổ trọng tài cho trận đấu, Đăng kí danh sách cầu thủ cho trận đấu, Lên lịch thi đấu

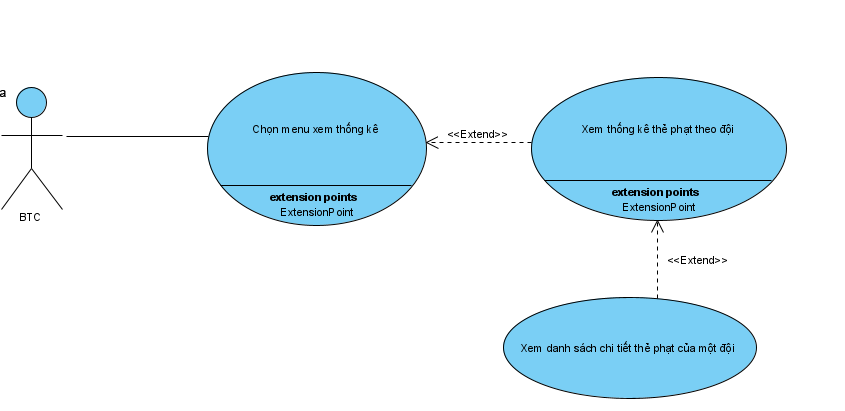


Hình 3.1: Biểu đồ use case tổng quan

Mô tả các use case như sau:

* Modul "Lên lịch thi đấu" cho phép BTC lên lịch thi đấu cho từng trận đấu
* Modul "Đăng kí danh sách cầu thủ cho trận đấu" cho phép BTC đăng kí danh sách cầu thủ thi đấu cho từng trận đấu
* Modul "Phân công tổ trọng tài cho trận đấu" cho phép BTC đăng kí các tổ trọng tài cho từng trận đấu
* Modul "Cập nhật kết quả thi đấu" cho phép BTC cập nhật kết quả cho từng trận đấu
* Modul "Thanh toán tiền công cho trọng tài" cho phép BTC thanh toán tiền công cho các trọng tài theo đợt
* Modul "Kí hợp đồng bản quyền truyền hình" cho phép BTC thêm hợp đồng truyền hình với các đối tác
* Modul "Thanh toán tiền bản quyền truyền hình" cho phép BTC thanh toán tiền bản quyền truyền hình với các đội bóng
* Modul "Xem BXH đội bóng" cho phép BTC thống kê và sắp xếp các đội bóng theo thành tích tại giải
* Modul "Xem BXH các cầu thủ ghi nhiều bàn thắng" cho phép BTC thống kê và sắp xếp các cầu thủ theo số bàn thắng để trao giải Goldshoes
* Modul "Thống kê thẻ phạt các đội bóng" cho phép BTC thống kê và sắp xếp các đội theo số thẻ phạt để trao giải Fairplay
* Modul "Thống kê các sân theo số lượng khán giả" cho phép BTC thống kê và sắp xếp các sân theo số lượng khán giả đến sân

### Use chi tiết Thống kê thẻ phạt các đội bóng

****

Hình 3.2: Biểu đồ use case chi tiết của Thống kê thẻ phạt

Trong Hình 3.2. Các UC được mô tả như sau:

* Xem menu thống kê -> đề xuất UC xem menu thống kê.
* Xem thống kê thẻ phạt theo đội -> đề xuất UC xem thống kê thẻ phạt
* Xem danh sách chi tiết thẻ phạt của một đội -> đề xuất UC xem danh sách chi tiết thẻ phạt của một đội
* Các giao diện thống kê lần lượt theo dạng: giao diện sau là tùy chọn từ giao diện trước. Do đó, chúng có quan hệ mở rộng lần lượt cái sau từ cái trước

Như vậy, biểu đồ UC chi tiết cho chức năng thống kê sản phẩm theo doanh thu được trình bày như trong hình vẽ. Trong đó, các UC chi tiết được mô tả như sau:

* Xem TKthẻ phạt theo đội : UC này cho phép BTC xem thống kê thẻ phạt của các đội.
* Xem danh sách chi tiết thẻ phạt của mỗi một đội: UC này cho phép BTC xem chi tiết thẻ phạt của một đội được chọn

## PHÂN TÍCH YÊU CẦU

## Viết kịch bản

Lưu ý tránh sai sót trong kịch bản:

* + - * Mỗi use case chính trong biểu đồ use case tổng quan thường có một kịch bản chuẩn và có thể có nhiều kịch bản ngoại lệ.
      * Use case có bao nhiêu actor thì kịch bản cũng có bấy nhiêu actor
      * Kịch bản càng chi tiết càng tốt: thông tin hệ thống hiện lên, thông tin người dùng nhập vào...

### Kịch bản cho Thống kê thẻ phạt

|  |  |
| --- | --- |
| Use case | Thống kê thẻ phạt |
| Actor | Quản trị hệ thống (QTHT) |
| Tiền điều kiện | Quản trị viên đã đăng nhập thành công |
| Hậu điều kiện |  |
| Kịch bản chính | 1. Sau khi đăng nhập, từ giao diện chính, QTHT chọn chức năng Thống kê thẻ phạt. 2. Hệ thống lấy danh sách thống kê tổng số thẻ đỏ, thẻ vàng, tổng số lần phạm lỗi của các đội bóng. Sau đón hiển thị giao diện thống kê. |
| Ngoại lệ | 2. Không có danh sách thẻ phạt.  3. Không kết nối được đến CSDL. |

### Kịch bản cho module Thống kê chi tiết thẻ phạt từng đội bóng

|  |  |
| --- | --- |
| Use case | Thống kê chi tiết thẻ phạt từng đội bóng |
| Actor | Quản trị hệ thống (QTHT) |
| Tiền điều kiện | Quản trị viên đã đăng nhập thành công |
| Hậu điều kiện |  |
| Kịch bản chính | 1. Sau khi đăng nhập, từ giao diện chính, QTHT chọn chức năng Thống kê thẻ phạt. 2. Hệ thống lấy danh sách thống kê tổng số thẻ đỏ, thẻ vàng, tổng số lần phạm lỗi của các đội bóng. Sau đón hiển thị giao diện thống kê. |
| Ngoại lệ | 1. Không có danh sách thẻ phạt. 2. Không kết nối được đến CSDL. |

## Trích lớp thực thể

Tóm tắt các bước trong phương pháp trích danh từ để trích lớp thực thể cho hệ thống:

* + - * Bước 1: Mô tả ngắn gọn nhưng phải đầy đủ hệ thống trong một đoạn văn. Có thể mô tả lần lượt các chức năng của hệ thống theo từng người dùng để tránh bị sót. Có thể thay thế bước này bằng cách tổng hợp tất cả các kịch bản (bao gồm kịch bản chuẩn và kịch bản ngoại lệ) của tất cả các use case của hệ thống.
      * Bước 2: Trích các danh từ xuất hiện trong đoạn văn của bước 1. Mỗi danh từ xuất hiện chỉ cần tính một lần.
      * Bước 3: Đánh giá các danh từ.
        + Một là đề xuất danh từ thành một lớp thực thể.
        + Hai là đề xuất danh từ thành thuộc tính của lớp nào đó.
        + Ba là loại bỏ danh từ đó vì quá chung chung, trừu tượng, hoặc ngoài pham vi quản lí của hệ thống.
      * Bước 4: Xác định quan hệ số lượng giữa các thực thể.
        + Nếu quan hệ 1-1 thì có thể giữ nguyên hoặc gộp lại.
        + Nếu quan hệ 1-n thì giữ nguyên.
        + Nếu quan hệ là n-n thì phải đề xuất các lớp thực thể trung gian để tách thành ít nhất 2 quan hệ 1-n.
      * Bước 5: Xác định quan hệ đối tượng giữa các thực thể: quan hệ kế thừa, hợp thành (aggregation), gắn chặt (composition), liên kết (association), phụ thuộc (dependence)...

Lưu ý khi vẽ biểu đồ lớp thực thể pha phân tích:

* + - * Tên lớp, tên thuộc tính có thể dùng ngôn ngữ tự nhiên vẫn được, tuy nhiên, nên đặt tên theo chuẩn code convention để vào pha thiết kế đỡ phải thay đổi.
      * Thuộc tính chưa cần kiểu dữ liệu
      * Các lớp thực thể chưa cần thuộc tính id trong pha phân tích.

Áp dụng cho module thống kê thẻ phạt các đội bóng, các bước được thực hiện như sau:

### Bước 1: Mô tả hệ thống trong một đoạn văn

Hệ thống là một trang web hỗ trợ ban tổ chức thống kê tổng số thẻ phạt của cá đội bóng trong một giải đấu bóng đá. Trong đó, quản trị hệ thống (QTHT) được thực hiện chức năng thống kê tổng số thẻ phạt của các đội bóng và xem chi tiết thống kê của từng đội. QTHT quản lý các vấn đề liên quan đến giải đấu như: đội bóng, cầu thủ, giải đấu, vòng đấu, sân đấu, trận đấu, trọng tài với các thao tác thêm, sửa, xóa. Xem kết quả của các trận đấu.

### Bước 2: Trích các danh từ xuất hiện trong đoạn văn

* + - * Các danh từ liên quan đến người: ban tổ chức, cầu thủ, trọng tài
      * Các danh từ liên quan đến vật: đội bóng, giải đấu, vòng đấu, sân đấu, trận đấu
      * Các danh từ liên quan đến thông tin: thẻ phạt, giải đấu, đội bóng, kết quả.

### Bước 3: Đánh giá và lựa chọn các danh từ làm lớp thực thể hoặc thuộc tính

* + - * Các danh từ liên quan đến người:
        + Ban tổ chức → BanToChuc: mã, số điện thoại, email.
        + Cầu thủ → CauThu: mã, tên, năm sinh, vị trí, chân thuận, chiều cao, cân nặng, đặc điểm.
        + Trọng tài → TrongTai: mã, tên, số điện thoại, địa chỉ.
      * Các danh từ liên quan đến vật:
        + Giải đấu → GiaiDau: mã, tên giải đấu.
        + Đội bóng → DoiBong: mã, tên, trang phục truyền thống, mô tả
        + Vòng đấu → VongDau: mã, tên vòng đấu
        + Sân đấu → mã, tên, địa chỉ
        + Trận đấu → ngày, giờ, danh sách trọng tài, danh sách đội bóng
        + Kết quả → số thẻ đỏ, số thẻ vàng, số pha phạm lỗi, ban tổ chức, trận đấu

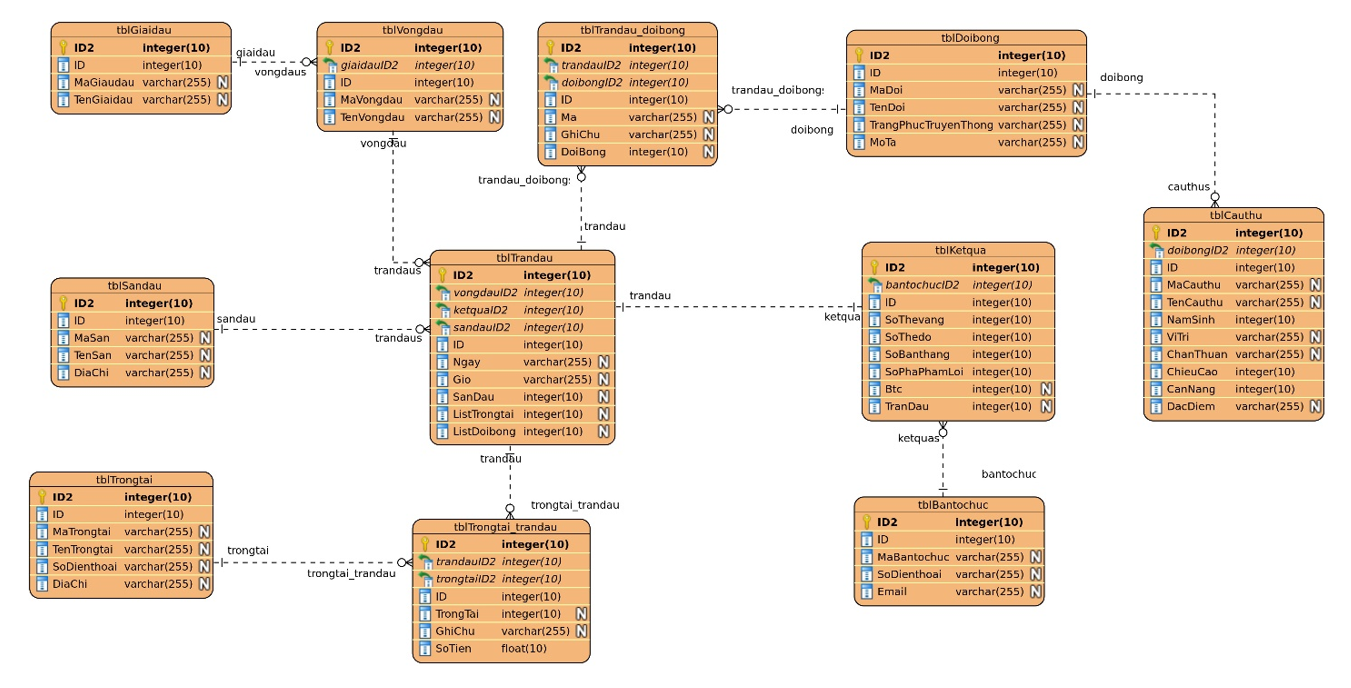
### Bước 4: Xác định quan hệ số lượng giữa các thực thể

* + - * Ban tổ chức tổ chức nhiều giải đấu → quan hệ giữa BanToChuc và GiaiDau là 1 – n
      * Giải đấu có nhiều vòng đấu → GiaiDau – VongDau 1 - n
      * Giải đấu có nhiều đội bóng → GiaiDau – DoiBong là 1 – n
      * Vòng đấu có nhiều trận đấu → VongDau – TranDau 1 – n
      * Trận đấu có nhiều đội bóng, nhiều (2) đội bóng trong một trận đấu → TranDau – DoiBong n – n
      * Trận đấu có nhiều trọng tài, trọng tài bắt nhiều trận đấu → TranDau – trongTai là n - n
      * Đội bóng có nhiều cầu thủ → DoiBoing – CauThu 1 – n
      * Kết quả của nhiều trận đấu → KetQua – TranDau 1 - n

### Bước 5: Xác định quan hệ đối tượng giữa các thực thể

* + - * Giải đấu là thành phần của Ban tổ chức.
      * Vòng đấu là thành phần của Giải đấu.
      * Trân đấu là thành phần của Vòng đấu.
      * Đội bóng là thành phần của Trận đấu.
      * Trọng tài là thành phần của Trận đấu
      * Cầu thủ là thành phần của Đội bóng.
      * Kết quả của nhiều trận đấu.
      * Một trận đấu có nhiều trọng tài và một trọng tài có thể bắt nhiều trận đấu

Kết quả thu được biểu đồ lớp thực thể pha phân tích toàn hệ thống như trong Hình 3.5.

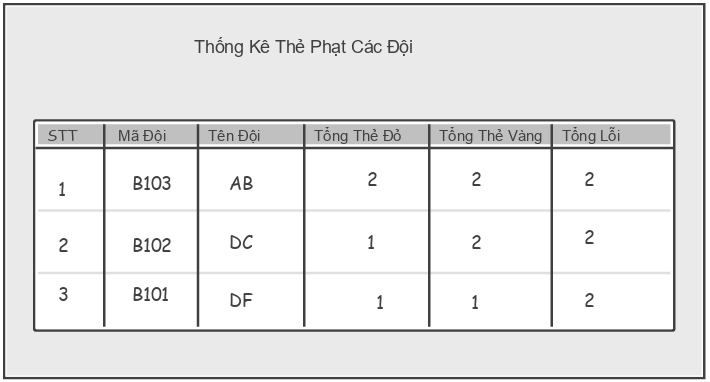


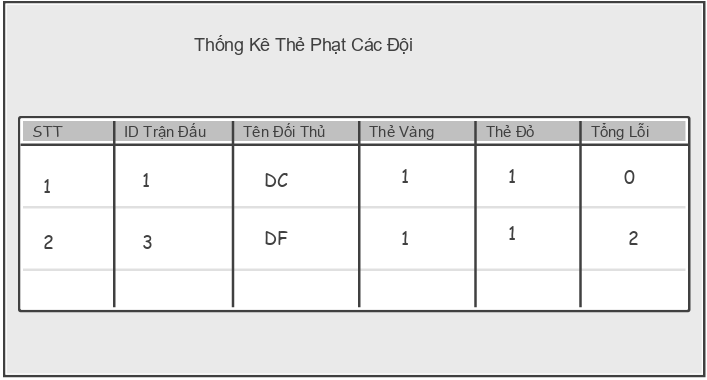
Hình 3.5: Biểu đồ lớp thực thể pha phân tích của hệ thống

## Thiết kế giao diện cho module Thống kê số thẻ phạt các đội bóng

* Giao diện cho module Thống kê số thẻ phạt các đội bóng:
* GDTrangchu: Giao diện trang chủ
* GDThongke: Giao diện thống kê
* GDThongkethephat: Giao diện thống kê thẻ phạt
* GDThongkechitiet: Giao diện thống kê thẻ phạt chi tiết một đội bóng.







## Trích các lớp biên và điều khiển

Tóm tắt các bước thực hiện để vẽ biểu đồ lớp pha phân tích:

* + - * Bước 1: Mỗi giao diện xuất hiện có thể đề xuất thành một lớp biên.
      * Bước 2: Với mỗi giao diện, xem xét cần các thành phần nào để nhận dữ liệu vào, hiện dữ liệu lên, hay submit dữ liệu vào hệ thống. Mỗi thành phần như vậy đề xuất thành một thuộc tính của giao diện, đặt tên với các tiền tố tương ứng:
        + in: tiền tố của tên thuộc tính nhận dữ liệu vào (ô text, ô chọn. )
        + out: tiền tố của tên thuộc tính hiện dữ liệu lên (bảng biểu, thông báo...)
        + sub: tiền tố của tên thuộc tính để submit dữ liệu (nút nhấn, liên kết...)
      * Bước 3: Với mỗi chức năng phải thực hiện tầng dưới tầng giao diện, thực hiện đề xuất phương thức:
        + Đề xuất tên phương thức (có thể đặt tên gợi nhớ, và tuân thủ code convention)
        + Xác định tham số đầu vào là gì, tham số đầu ra là gì.
        + Đề xuất gán hành động tương ứng với chức năng này cho lớp thực thể nào. Theo nguyên tắc:

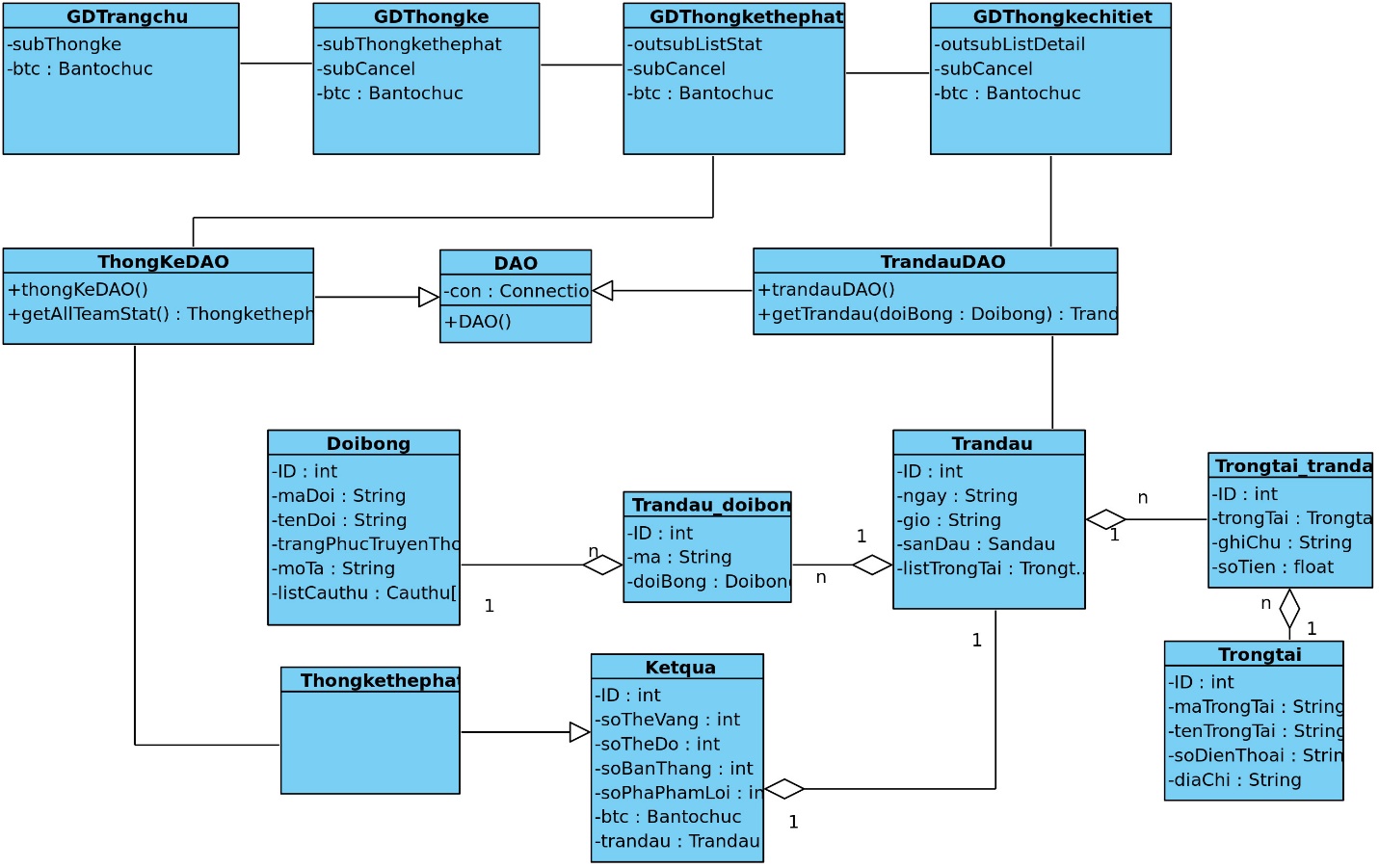
Nếu tham số đầu ra liên quan lớp thực thể nào thì gán phương thức này cho lớp đó

Nếu không, xét tham số đầu vào. Nếu tham số đầu vào chỉ liên quan đến 1 thực thể thì gán phương thức này cho thực thể đó. Nếu tham số vào liên quan đến hai hay nhiều thực thể, thì tìm thực thể nào nhỏ nhất mà chứa được càng nhiều càng tốt các tham số đầu vào, gán phương thức này cho lớp thực thể đó.

* + - * Vẽ biểu đồ lớp pha phân tích cho từng module.

### Phân tích tĩnh Thống kê tẻ phạt các đội

* + Tầng giao diện gồm các trang jsp: GDTrangchu, GDThongke, GDThongkethephat, GDThongkechitiet
  + Tầng xử lý truy cập dữ liệu: DAO
    - ThongkeDAO: findPenatyCardsOfAllTeams()
    - TranDauDoiBongDAO: findAllPenaltyCardsByDoiBongId ()
  + Các lớp thực thể liên quan: Trandau, Thongkethephat, Doibong, Cauthu, Trongtai, Trongtai\_trandau, Trandau\_doibong



Hình 3.6: Biểu đồ lớp phân tích module thống kê thẻ phạt

## Phân tích hoạt động

Tóm tắt các bước thực hiện để vẽ biểu đồ trạng thái của một modul/chức năng:

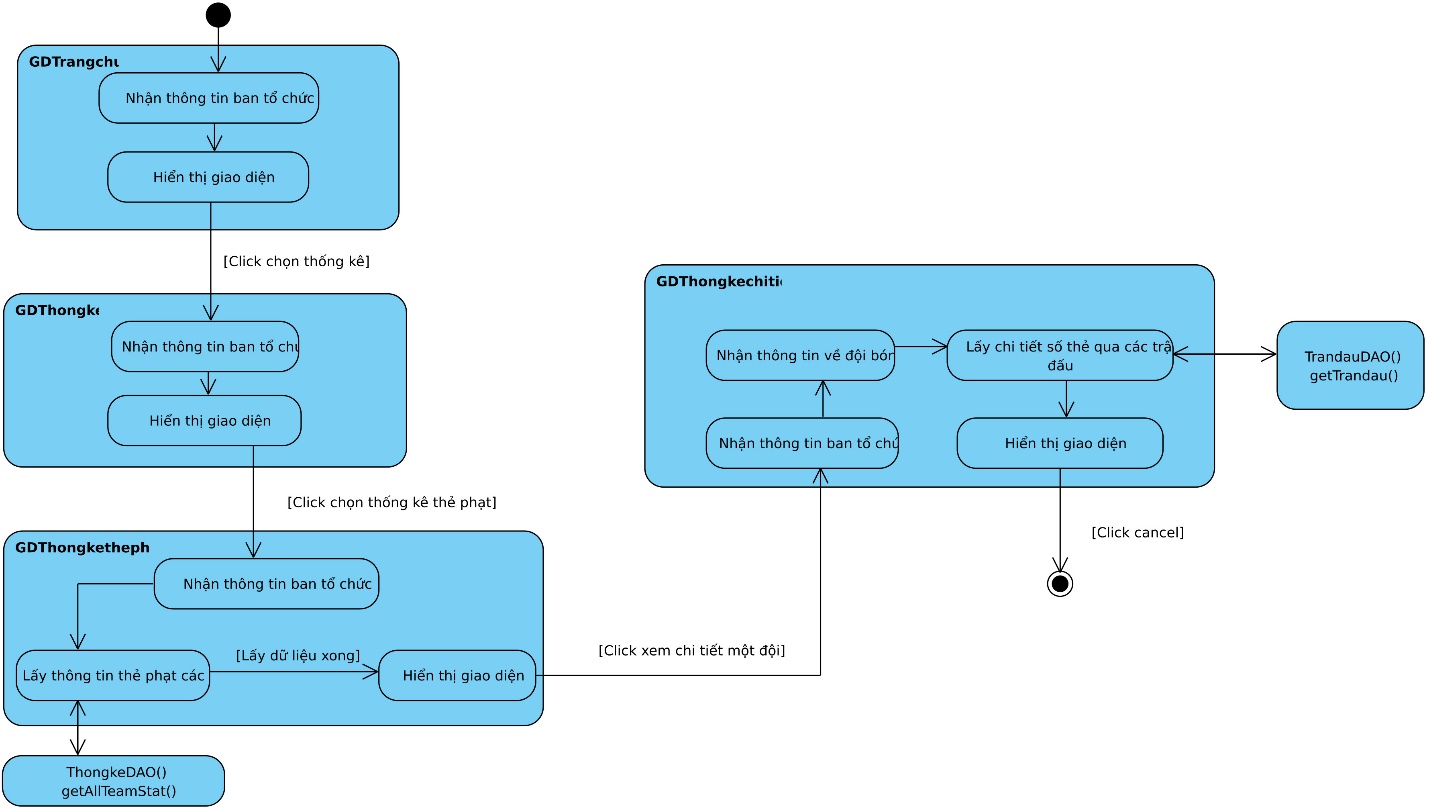
* + - * Bước 1: Mỗi trạng thái chờ tương ứng với mỗi lần hệ thống hiển thị 1 giao diện để chờ tương tác của người dùng
      * Bước 2: Điều kiện chuyển trạng thái tương ứng với hành động của người dùng trên giao diện tương ứng.

Tóm tắt các bước thực hiện để vẽ biểu đồ tuần tự/giao tiếp pha phân tích:

* + - * Bước 1: Viết kịch bản phiên bản 2 từ biểu đồ lớp của modul: Mỗi bước là một hành động của một lớp nào đó trong biểu đồ lớp, hoặc là của tác nhân.
      * Bước 2: Vẽ biểu đồ giao tiếp cho kịch bản chuẩn: các bước tương ứng với các bước trong kịch bản phiên bản 2. Quan hệ giữa các lớp phải thống nhất với quan hệ giữa chúng trong biểu đồ lớp của module.
      * Bước 3: chuyển hóa biểu đồ giao tiếp thành biểu đồ tuần tự cho kịch bản chuẩn. Kiểm tra lại các bước đúng tuần tự và logic nghiệp vụ.

### Module Thống kê thẻ phạt

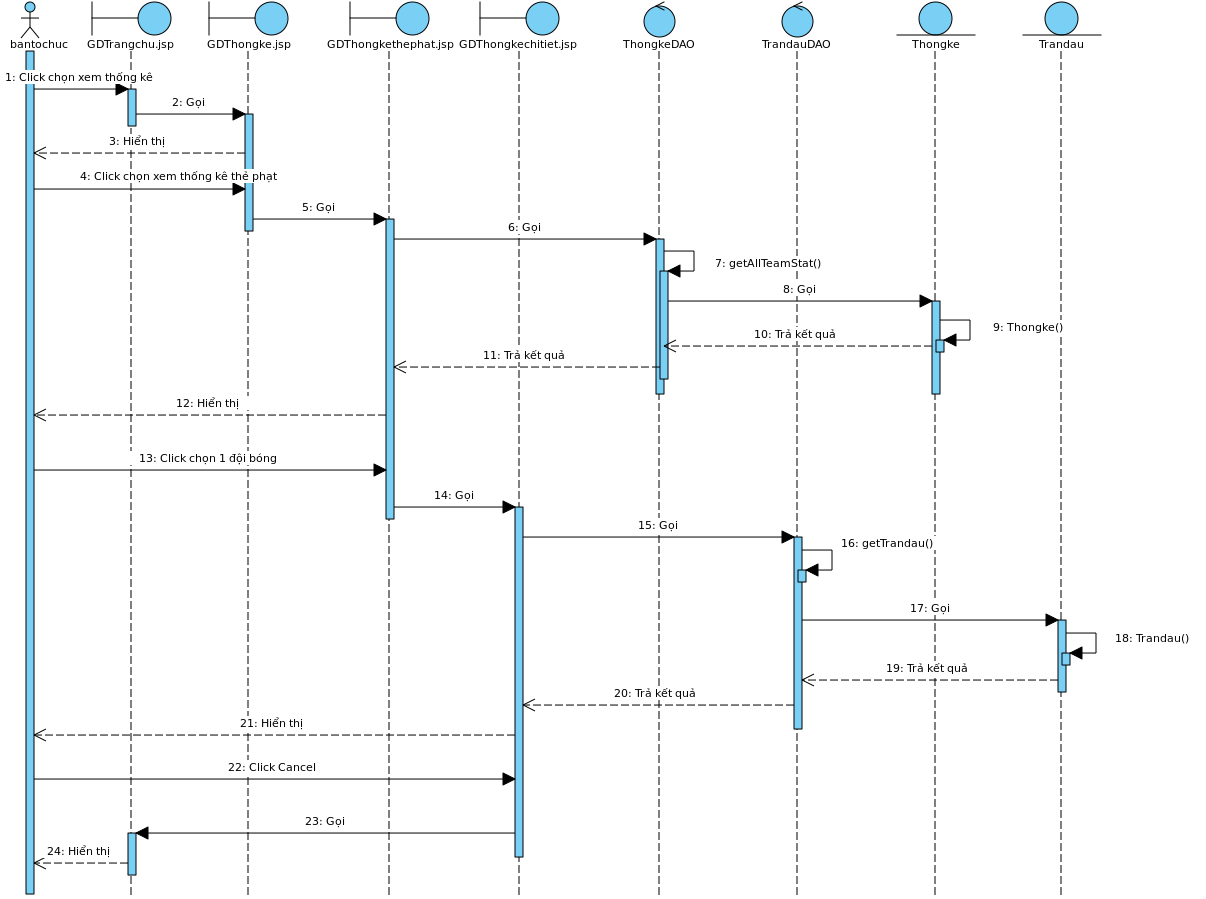
* + - Xây dựng biểu đồ hoạt động: Mỗi hành động tương ứng với một phương thức đã được thiết kế trong biểu đồ lớp
    - Tại giao diện trang chủ: GDTrangchu.jsp
      * Nhận thông tin ban tổ chức
      * Hiển thị giao diện
    - Tại giao diện thống kê: GDThongke
      * Nhận giao diện ban tổ chức
      * Hiển thị giao diện
    - Tại giao diện thống kê thẻ phạt: GDThongkethephat:
      * Nhận thông tin lễ tân
      * Lấy thông tin các đội bóng kèm số thẻ phạt: findPenaltyCardsOfAllTeams () tại lớp ThongkeDAO
      * Hiển thị giao diện
    - Tại giao diện thống kê thẻ phạt chi tiết: GDThongkechitiet
      * Nhận thông tin lễ tân
      * Nhận thông tin id đội bóng.
      * Tìm kiếm thông tin thẻ phạt của đội bóng ở các trận đấu: findAllPenaltyCardsByDoiBongId() trong lớp TranDauDoiBongDAO.
      * Hiển thị giao diện



Hình 3.9: Biểu đồ chuyển trạng thái của modul thống kê thẻ phạt

Kịch bản v.3:

1. Ban tổ chức chọn chức năng xem thống kê sau khi đã đăng nhập vào hệ thống, ban tổ chức muốn xem thông kê.
2. Lớp GDTrangchu.jsp gọi lớp GDThongkethephat.jsp
3. Lớp GDThongkethephat.jsp gọi lớp ThongkeDAO yêu cầu tìm tất cả số thẻ phạt của các đội bóng.
4. Lớp ThongkeDAO gọi hàm findPenaltyCardsOfAllTerams().
5. Hàm findPenaltyCardsOfAllTerams() gọi lớp Thongke yêu cầu đóng gói thông tin.
6. Lớp Thongke đóng gói thông tin
7. Lớp Thongke trả kết quả cho hàm findPenaltyCardsOfAllTerams()
8. Hàm findPenaltyCardsOfAllTerams() trả kết quả cho lớp GDThongkethephat.jsp
9. Lớp Thongkethephat.jsp hiển thị kết quả cho ban tổ chức.
10. Ban tổ chức click chọn một đội bóng muốn xem chi tiết số thẻ phạt.
11. Lớp GDThongkethephat.jsp gọi lớp GDThongkechitiet.jsp
12. Lớp GDThongkechitiet.jsp gọi lớp TranDauDoiBongDAO yêu cầu lấy thông tin số thẻ phạt của đội bóng qua các trận đấu.
13. Lớp TranDauDoiBongDAO gọi hàm findAllPenaltyCardsByDoiBongId ().
14. Hàm findAllPenaltyCardsByDoiBongId () gọi lớp Trandau yêu cầu đóng gói thông tin.
15. Lớp Trandau đóng gói thông tin
16. Lớp Trandau trả kết quả cho hàm findAllPenaltyCardsByDoiBongId ().
17. Hàm findAllPenaltyCardsByDoiBongId () trả kết quả cho lớp GDThongkechitiet.jsp
18. Lớp GDThongkechitiet.jsp hiển thị kết quả với ban tổ chức.
19. Ban tổ chức click Cancel để quay lại trang chủ sau khi xem xong.
20. Lớp GDThongkechitiet.jsp gọi lớp GDTrangchu.jsp
21. Lớp GDTrangchu.jsp hiển thị



*Sơ đồ tuần tự pha thiết kế module Thống kê thẻ phạt các đội*

## CHECKLIST

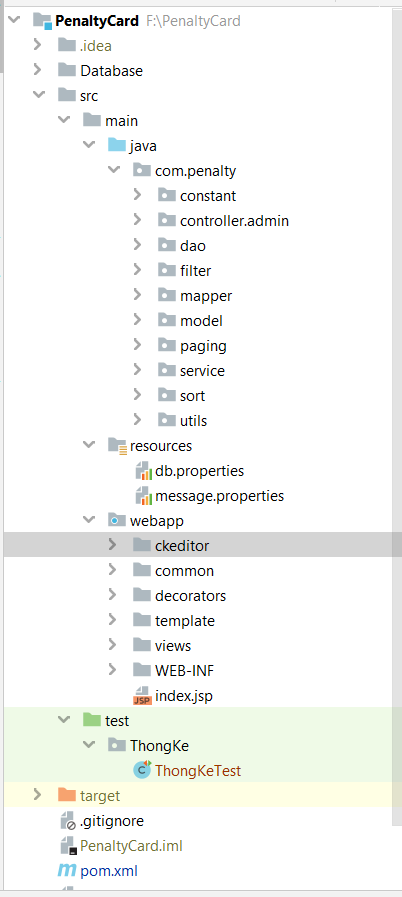
1. Lập danh sách glossary list cho ứng dụng của nhóm.
2. Mô tả hệ thống bằng ngôn ngữ tự nhiên.
3. Vẽ biểu đồ use case toàn hệ thống
4. Vẽ biểu đồ use case cho modul cá nhân
5. Viết kịch bản cho modul cá nhân
6. Trích lớp thực thể và vẽ biểu đồ lớp thực thể cho toàn hệ thống
7. Trích và vẽ biểu đồ lớp cho modul
8. Vẽ biểu đồ trạng thái cho modul
9. Viết kịch bản v.2 và vẽ biểu đồ giao tiếp cho modul

# CHƯƠNG 5: CÀI ĐẶT HỆ THỐNG

## TỔ CHỨC DỰ ÁN

Dự án trong IntelliJ IDEA được tổ chức như Hình 5.1:

* + - Các lớp thực thể (model), điều khiển (controller), lớp thao tác với cơ sở dữ liệu (dao), lớp thực thi nghiệp vụ (service), ... để trong thư mục /src/main/java/, chia nhỏ thành các package con tương ứng: model, controller, dao, service, ...
    - Thư mục /resources/ chứa các cấu hình kết nối đến CSDL và các message thông báo.
    - Các trang jsp để trong thư mục /webapp/views/.
    - Thư mục /test/ chứa các unit test.
    - Project sử dụng Maven nên các thư viện kết nối CSDL, Junit và các thư viện khác thì sẽ được khai báo dependencies trong file pom.xml.



Hình 5.1: Tổ chức thư mục dự án web trong IntelliJ IDEA

## CÀI ĐẶT CÁC MODULE

Nội dung phần này sẽ trình bày cài đặt 2 chức năng: Lấy danh sách thống kê tổng số thẻ phạt của các đội bóng, Xem chi tiết thống kê thẻ phạt của từng đội qua mỗi trận đấu. Tổng quan:

* + - Phần thực thể (model) không trình bày code ở đây.
    - Chức năng Thống kê tổng số thẻ phạt và Xem chi tiết thống kê của từng đội chỉ được cài đặt phần DAO , phần JSP và giao diện demo.
    - Phần JSP có sử dụng template mẫu được xử lý html/css để minh họa nghiệp vụ. Và sử dụng JSTL (JSP Standard Tag Library) để thao tác với dữ liệu được gửi lên view.
    - Người đọc cần chỉnh sửa thông tin kết nối vào CSDL (driverName, url, username, password) trong lớp DAO cho phù hợp với cấu hình CSDL trên máy mình.
    - Nếu người đọc đổi tên bảng/tên cột trong CSDL thì phải cập nhật tương ứng trong các câu lệnh SQL.

## Cấu hình và kết nối đến cơ sở dữ liệu

***File db.properties***

**driverName** = **com.mysql.cj.jdbc.Driver**

**url** = **jdbc:mysql://localhost:3306/penalty  
user** = **root  
password** = **1234**

***Lớp GenericDAO.java (Interface gồm các thao tác đến CSDL: select, insert, update, delete)***

**public interface** GenericDAO<T> {  
 <T> List<T> query(String sql, RowMapper<T> rowMapper, Object... parameters);  
 **void** update (String sql, Object... parameters);  
 **int** insert (String sql, Object... parameters);  
 **int** count(String sql, Object... parameters);  
}

***Lớp AbstractDAO.java (Class kết nối đến CSDL và thực thi các phương thức từ GenericDAO.java)***

**public class** AbstractDAO<T> **implements** GenericDAO<T> {

**private** ResourceBundle **resourceBundle** = ResourceBundle.*getBundle*(**"db"**);  
  
**private** Connection getConnection() {  
 **try** {  
 Class.*forName*(**resourceBundle**.getString(**"driverName"**));  
 String url = **resourceBundle**.getString(**"url"**);  
 String user = **resourceBundle**.getString(**"user"**);  
 String password = **resourceBundle**.getString(**"password"**);  
 **return** DriverManager.*getConnection*(url, user, password);  
 } **catch** (ClassNotFoundException | SQLException e) {  
 **return null**;  
 }  
}

@Override  
**public** <T> List<T> query(String sql, RowMapper<T> rowMapper, Object... parameters) {  
 List<T> results = **new** ArrayList<>();  
 Connection connection = **null**;  
 PreparedStatement statement = **null**;  
 ResultSet resultSet = **null**;  
 **try** {  
 connection = getConnection();  
 statement = connection.prepareStatement(sql);  
 setParameter(statement, parameters);  
 resultSet = statement.executeQuery();  
 **while** (resultSet.next()) {  
 results.add(rowMapper.mapRow(resultSet));  
 }  
 **return** results;  
 } **catch** (SQLException e) {  
 **return null**;  
 } **finally** {  
 **try** {  
 **if** (connection != **null**) {  
 connection.close();  
 }  
 **if** (statement != **null**) {  
 statement.close();  
 }  
 **if** (resultSet != **null**) {  
 resultSet.close();  
 }  
 } **catch** (SQLException e) {  
 **return null**;  
 }  
 }  
}

**private void** setParameter(PreparedStatement statement, Object... parameters) {  
 **try** {  
 **for** (**int** i = 0; i < parameters.**length**; i++) {  
 Object parameter = parameters[i];  
 **int** index = i + 1;  
 **if** (parameter **instanceof** Long) {  
 statement.setLong(index, (Long) parameter);  
 } **else if** (parameter **instanceof** String) {  
 statement.setString(index, (String) parameter);  
 } **else if** (parameter **instanceof** Integer) {  
 statement.setInt(index, (Integer) parameter);  
 } **else if** (parameter **instanceof** Timestamp) {  
 statement.setTimestamp(index, (Timestamp) parameter);  
 } **else if** (parameter **instanceof** Double) {  
 statement.setDouble(index, (Double) parameter);  
 }  
 }  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
}  
  
@Override  
**public void** update(String sql, Object... parameters) {  
 Connection connection = **null**;  
 PreparedStatement statement = **null**;  
 **try** {  
 connection = getConnection();  
 connection.setAutoCommit(**false**);  
 statement = connection.prepareStatement(sql);  
 setParameter(statement, parameters);  
 statement.executeUpdate();  
 connection.commit();  
 } **catch** (SQLException e) {  
 **try** {  
 connection.rollback();  
 } **catch** (SQLException e1) {  
 e1.printStackTrace();  
 }  
 } **finally** {  
 **try** {  
 **if** (connection != **null**) {  
 connection.close();  
 }  
 **if** (statement != **null**) {  
 statement.close();  
 }  
 } **catch** (SQLException e2) {  
 e2.printStackTrace();  
 }  
 }  
}  
  
@Override  
**public int** insert(String sql, Object... parameters) {  
 Connection connection = **null**;  
 PreparedStatement statement = **null**;  
 ResultSet resultSet = **null**;  
 **try** {  
 **int** id = 0;  
 connection = getConnection();  
 connection.setAutoCommit(**false**);  
 statement = connection.prepareStatement(sql, Statement.***RETURN\_GENERATED\_KEYS***);  
 setParameter(statement, parameters);  
 statement.executeUpdate();  
 resultSet = statement.getGeneratedKeys();  
 **if** (resultSet.next()) {  
 id = resultSet.getInt(1);  
 }  
 connection.commit();  
 **return** id;  
 } **catch** (SQLException e) {  
 **if** (connection != **null**) {  
 **try** {  
 connection.rollback();  
 } **catch** (SQLException e1) {  
 e1.printStackTrace();  
 }  
 }  
 } **finally** {  
 **try** {  
 **if** (connection != **null**) {  
 connection.close();  
 }  
 **if** (statement != **null**) {  
 statement.close();  
 }  
 **if** (resultSet != **null**) {  
 resultSet.close();  
 }  
 } **catch** (SQLException e2) {  
 e2.printStackTrace();  
 }  
 }  
 **return** 0;  
}  
  
@Override  
**public int** count(String sql, Object... parameters) {  
 Connection connection = **null**;  
 PreparedStatement statement = **null**;  
 ResultSet resultSet = **null**;  
 **try** {  
 **int** count = 0;  
 connection = getConnection();  
 statement = connection.prepareStatement(sql);  
 setParameter(statement, parameters);  
 resultSet = statement.executeQuery();  
 **while** (resultSet.next()) {  
 count = resultSet.getInt(1);  
 }  
 **return** count;  
 } **catch** (SQLException e) {  
 **return** 0;  
 } **finally** {  
 **try** {  
 **if** (connection != **null**) {  
 connection.close();  
 }  
 **if** (statement != **null**) {  
 statement.close();  
 }  
 **if** (resultSet != **null**) {  
 resultSet.close();  
 }  
 } **catch** (SQLException e) {  
 **return** 0;  
 }  
 }  
}

## Chức Thống kê tổng số thẻ phạt của các đội bóng

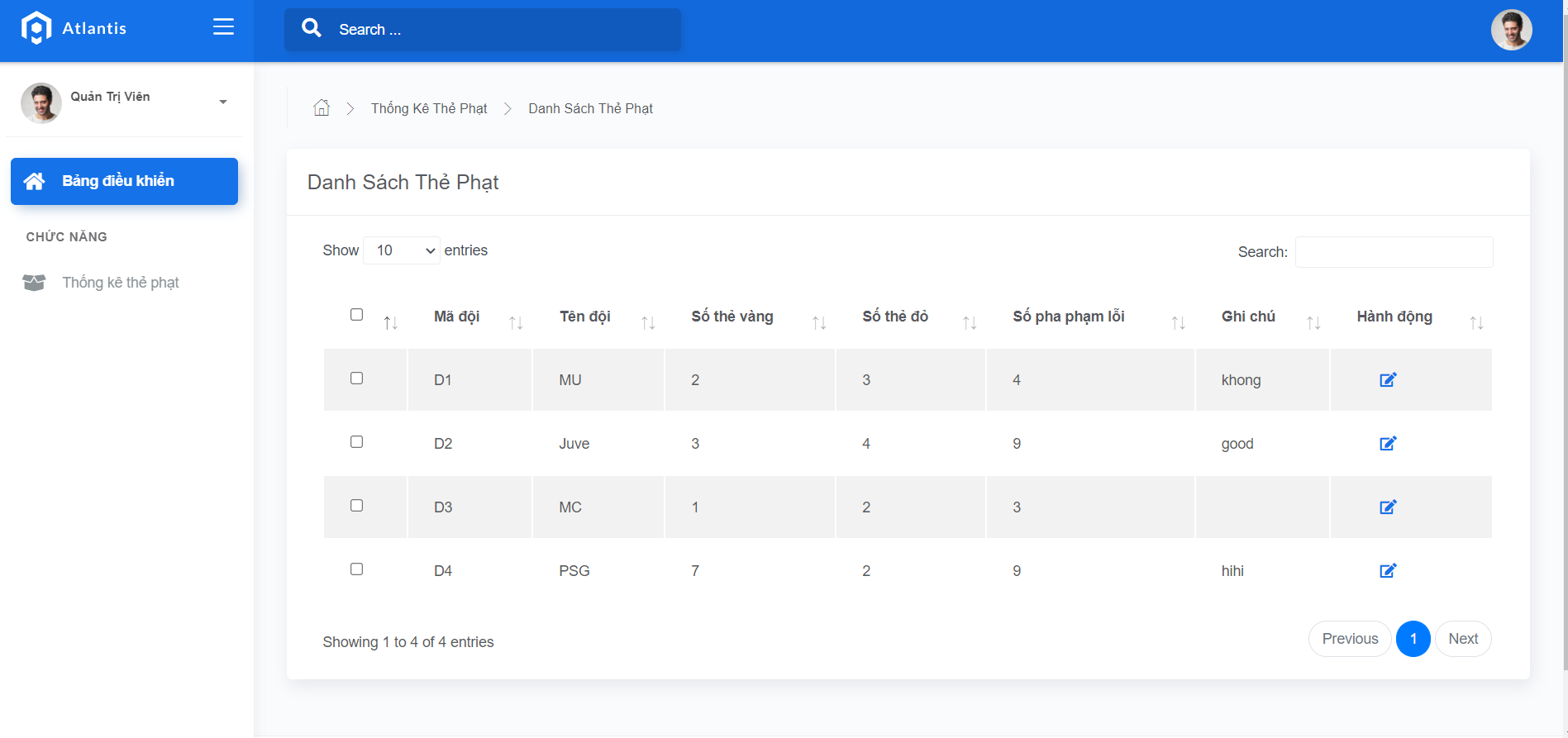
### Lớp TranDauDoiBongDAO.java

**public class** TranDauDoiBongDAO **extends** AbstractDAO<TranDauDoiBong> **implements** ITranDauDoiBongDAO {  
 @Override  
 **public** List<TranDauDoiBong> findAll() {  
 String sql = **" select tddb.id, tddb.tran\_dau\_id, tddb.doi\_bong\_id, db.ma\_doi, db.ten\_doi, tddb.ghi\_chu, "** +  
 **" sum(tddb.so\_the\_vang) so\_the\_vang, sum(tddb.so\_the\_do) so\_the\_do, sum(tddb.so\_pha\_pham\_loi) so\_pha\_pham\_loi "** +  
 **" from tran\_dau\_doi\_bong tddb join doi\_bong db on tddb.doi\_bong\_id = db.id group by db.id "**;  
 **return** query(sql, **new** TranDauDoiBongMapper());  
 }  
  
 @Override  
 **public** TranDauDoiBong findOne(**int** id) {  
 String sql = **" select tddb.id, tddb.tran\_dau\_id, tddb.doi\_bong\_id, db.ma\_doi, db.ten\_doi, tddb.ghi\_chu, "** +  
 **" tddb.so\_the\_vang, tddb.so\_the\_do, tddb.so\_pha\_pham\_loi "** +  
 **" from tran\_dau\_doi\_bong tddb join doi\_bong db on tddb.doi\_bong\_id=db.id where db.id = ? "**;  
 List<TranDauDoiBong> tranDauDoiBongs = query(sql, **new** TranDauDoiBongMapper(), id);  
 **return** tranDauDoiBongs.isEmpty() ? **null** : tranDauDoiBongs.get(0);  
 }  
}

### Giao diện thống kê tổng số thẻ phạt

<%@**include file**="**/common/taglib.jsp**"%>  
<**c:url var="APIurl" value="/api-thong-ke"**/>  
<**c:url var ="ThongKeURL" value="/thong-ke"**/>  
<%@ **page language**="**java**" **contentType**="**text/html; charset=UTF-8**" **pageEncoding**="**UTF-8**"%>  
<!DOCTYPE **html**>  
<**html**>  
<**head**>  
 <**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"**>  
 <**title**>Thống Kê Thẻ Phạt</**title**>  
</**head**>  
  
<**body**>  
<**div class="content"**>  
 <**div class="page-inner"**>  
 <**div class="page-header"**>  
 <**ul class="breadcrumbs" style="margin-left**: 0;**"**>  
 <**li class="nav-home"**>  
 <**a href="**<**c:url value="/admin-home"**/> **"**>  
 <**i class="flaticon-home"**></**i**>  
 </**a**>  
 </**li**>  
 <**li class="separator"**>  
 <**i class="flaticon-right-arrow"**></**i**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a href="**<**c:url value="/thong-ke?type=list"**/> **"**>Thống Kê Thẻ Phạt</**a**>  
 </**li**>  
 <**li class="separator"**>  
 <**i class="flaticon-right-arrow"**></**i**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a href="#"**>Danh Sách Thẻ Phạt</**a**>  
 </**li**>  
 </**ul**>  
 </**div**>  
 <**div class="row"**>  
 <**div class="col-md-12"**>  
 <**div class="card"**>  
 <**div class="card-header"**>  
 <**div class="d-flex align-items-center"**>  
 <**h4 class="card-title"**>Danh Sách Thẻ Phạt</**h4**></**div**>  
 </**div**>  
 <**div class="card-body"**>  
 <**div class="table-responsive"**>  
 <**table id="add-row" class="display table table-striped table-hover"** >  
 <**thead**>  
 <**tr**>  
 <**th**><**input type="checkbox" class="checkAll" id="checkAll"**></**th**>  
 <**th**>Mã đội</**th**>  
 <**th**>Tên đội</**th**>  
 <**th**>Số thẻ vàng</**th**>  
 <**th**>Số thẻ đỏ</**th**>  
 <**th**>Số pha phạm lỗi</**th**>  
 <**th**>Ghi chú</**th**>  
 <**th**>Hành động</**th**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 <**c:forEach var="tddb" items="${**model.listResult**}"**>  
 <**tr**>  
 <**td**><**input type="checkbox" id="checkbox\_${**tddb.id**}" value="${**tddb.id**}"**></**td**>  
 <**td**>**${**tddb.maDoi**}**</**td**>  
 <**td**>**${**tddb.tenDoi**}**</**td**>  
 <**td**>**${**tddb.soTheVang**}**</**td**>  
 <**td**>**${**tddb.soTheDo**}**</**td**>  
 <**td**>**${**tddb.soPhaPhamLoi**}**</**td**>  
 <**td**>**${**tddb.ghiChu**}**</**td**>  
 <**td**>  
 <**c:url var="editURL" value="/thong-ke"**>  
 <**c:param name="type" value="edit"**/>  
 <**c:param name="id" value="${**tddb.id**}"**/>  
 <**c:param name="tranDauId" value="${**tddb.tranDauId**}"**/>  
 </**c:url**>  
 <**a href="${**editURL**}" data-toggle="tooltip" title=""class="btn btn-link btn-primary" data-original-title="Xem Chi Tiết"**>  
 <**i class="fa fa-edit"**></**i**>  
 </**a**>  
 </**td**>  
 </**tr**>  
 </**c:forEach**>  
 </**tbody**>  
 </**table**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

</**body**>  
</**html**>



## Chức năng Xem chi tiết thẻ phạt của từng đội

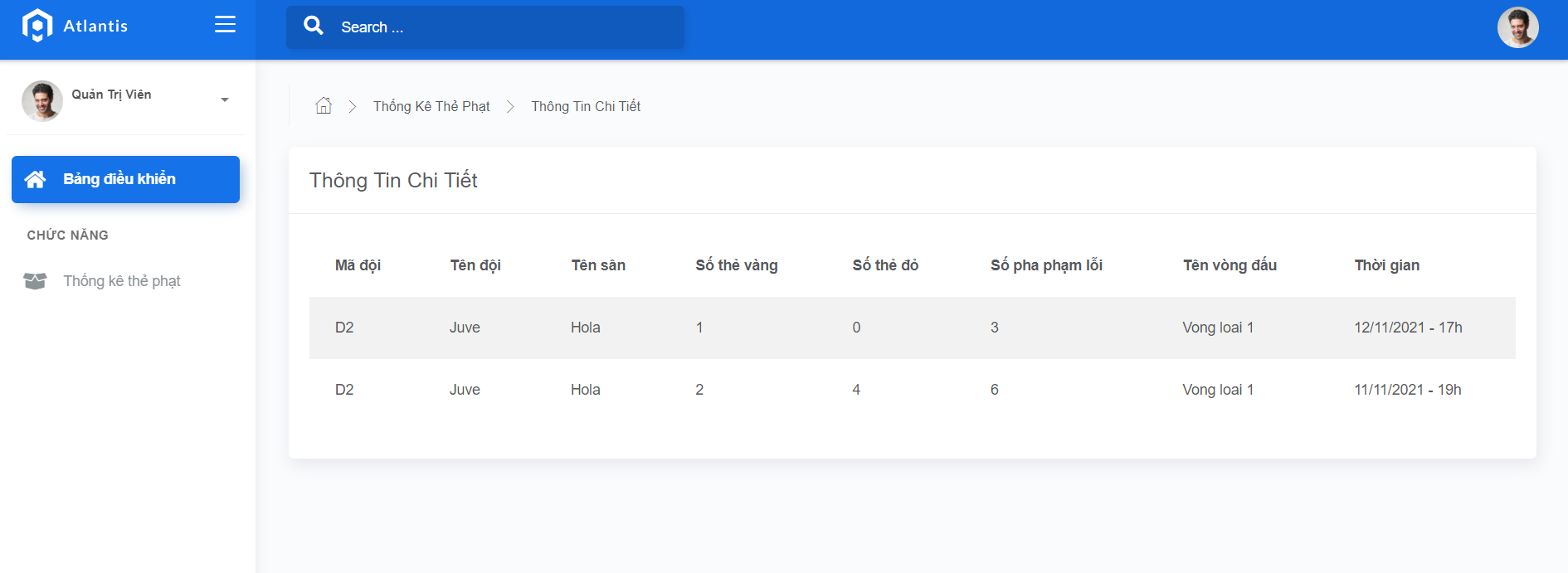
***Lớp ThongKeDAO.java***

**public class** ThongKeDAO **extends** AbstractDAO **implements** IThongKeDAO {  
  
 @Override  
 **public** List<ThongKe> findAllByDoiBongId(**int** doiBongId) {  
 String sql = **"select db.ma\_doi, db.ten\_doi, sd.ten\_san, tddb.so\_the\_vang, tddb.so\_the\_do, tddb.so\_pha\_pham\_loi, vd.ten\_vong\_dau,td.gio,td.ngay "** +  
 **" from doi\_bong db join tran\_dau\_doi\_bong tddb on db.id=tddb.doi\_bong\_id "** +  
 **" join tran\_dau td on td.id=tddb.tran\_dau\_id join san\_dau sd on td.san\_dau\_id=sd.id "** +  
 **" join vong\_dau vd on td.vong\_dau\_id = vd.id"** +  
 **" where db.id = ? "**;  
 **return** query(sql, **new** ThongKeMapper(), doiBongId);  
 }  
}

***Giao diện Xem chi tiết thẻ phạt của từng đội***

<%@ **page language**="**java**" **contentType**="**text/html; charset=UTF-8**" **pageEncoding**="**UTF-8**"%>  
<%@**include file**="**/common/taglib.jsp**"%>  
<**c:url var="APIurl" value="/api-thong-ke"**/>  
<**c:url var ="ThongKeURL" value="/thong-ke"**/>  
<**html**>  
<**head**>  
 <**title**>Thống Kê Chi Tiết</**title**>  
</**head**>  
<**body**>  
<**div class="content"**>  
 <**div class="page-inner"**>  
 <**div class="page-header"**>  
 <**ul class="breadcrumbs" style="margin-left**: 0;**"**>  
 <**li class="nav-home"**>  
 <**a href="**<**c:url value="/admin-home"**/> **"**>  
 <**i class="flaticon-home"**></**i**>  
 </**a**>  
 </**li**>  
 <**li class="separator"**>  
 <**i class="flaticon-right-arrow"**></**i**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a href="**<**c:url value="/ thong-ke?type=list"**/> **"**>Thống Kê Thẻ Phạt</**a**>  
 </**li**>  
 <**li class="separator"**>  
 <**i class="flaticon-right-arrow"**></**i**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a href="#"**>Thông Tin Chi Tiết</**a**>  
 </**li**>  
 </**ul**>  
 </**div**>  
 <**div class="row"**>  
 <**div class="col-md-12"**>  
 <**div class="card"**>  
 <**div class="card-header"**>  
 <**div class="d-flex align-items-center"**>  
 <**h4 class="card-title"**>Thông Tin Chi Tiết</**h4**>  
 </**div**>  
 </**div**>  
 <**div class="card-body"**>  
 <**div class="table-responsive"**>  
 <**table id="add-row" class="display table table-striped table-hover"** >  
 <**thead**>  
 <**tr**>  
 <**th**>Mã đội</**th**>  
 <**th**>Tên đội</**th**>  
 <**th**>Tên sân</**th**>  
 <**th**>Số thẻ vàng</**th**>  
 <**th**>Số thẻ đỏ</**th**>  
 <**th**>Số pha phạm lỗi</**th**>  
 <**th**>Tên vòng đấu</**th**>  
 <**th**>Thời gian</**th**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 <**c:forEach var="thongke" items="${**model.listResult**}"**>  
 <**tr**>  
 <**td**>**${**thongke.maDoi**}**</**td**>  
 <**td**>**${**thongke.tenDoi**}**</**td**>  
 <**td**>**${**thongke.tenSan**}**</**td**>  
 <**td**>**${**thongke.soTheVang**}**</**td**>  
 <**td**>**${**thongke.soTheDo**}**</**td**>  
 <**td**>**${**thongke.soPhaPhamLoi**}**</**td**>  
 <**td**>**${**thongke.tenVongDau**}**</**td**>  
 <**td**>**${**thongke.ngay**}** - **${**thongke.gio**}**h</**td**>  
 </**tr**>  
 </**c:forEach**>  
 </**tbody**>  
 </**table**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

</**body**>  
</**html**>



## KIỂM THỬ ĐƠN VỊ

Nội dung phần này sẽ trình bày kế hoạch kiểm thử đơn vị, các ca kiểm thử đơn vị cho lớp ThongKeDAO và TranDauDoiBongDAO.

## Xây dựng bộ test case cho kiểm thử đơn vị

Bộ test case cho kiểm thử đơn vị được xây dựng theo bảng sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Chức năng/ Use case** | **Lớp điều khiển** | **Phương thức** | **Trường hợp test** |
| 1 | Thống kê tổng số thẻ phạt | TranDauDoiBongDAO | findPenaltyCardsOfAllTeams() | Lấy danh sách thông tin thẻ phạt của các đội. |
| Danh sách thông tin thẻ phạt trống. |
| 2 | Xem thông tin chi tiết của từng đội | ThongKeDAO | findAllPenaltyCardsByDoiBongId() | Lấy danh sách thẻ phạt qua các trận của đội bóng có tồn tại. |
| Danh sách thẻ phạt trống của đội bóng có tồn tại. |
| Lấy danh sách của một đội bóng không tồn tại. |

## Cài đặt kiểm thử đơn vị

Việc thực hiện test đơn vị sẽ sử dụng Junit. Ngoại trừ các class interface, các class còn lại cần được thực hiện unit test. Mục tiêu của unit test là để đảm bảo từng unit (method, class) thực hiện đúng như trong thiết kế.

Quy trình tiến hành các bước như sau:

* + - * Bước 1: Với mỗi method/class, xây dựng test cases với lưu ý lựa chọn các input sao cho đạt mức phủ cấp 2 (phủ hết các nhánh trong source code), expected output thì cần căn cứ vào thiết kế, đặc tả để xác định, không căn cứ vào source code.
      * Bước 2: Viết script Junit
        + Với mỗi test cases, khai báo biến để lưu actual/expect result.
        + Gọi method cần test với test input ở trên.
        + Kiểm tra kết quả thực method với expected output.
        + Kiểm tra DB xem đã thay đổi như mong muốn chưa?
      * Bước 3: Báo cáo kết quả test, số lượng pass/fail và mô tả chi tiết trường hợp fail.

Lưu ý là cần thực hiện chuẩn bị môi trường (kết nối DB, dữ liệu mẫu) và cần Rollback sau khi thực hiện test để đảm bảo việc thực hiện các test cases khác, và sau này test lại sẽ không bị lỗi. .

***Lớp ThongKeTest.java***

**public class** ThongKeTest {  
  
 @Inject  
 **private** ThongKeDAO **thongKeDAO**;  
  
 @Inject  
 **private** TranDauDoiBongDAO **tranDauDoiBongDAO**;  
  
 @Before  
 **public void** before() {  
 **thongKeDAO** = **new** ThongKeDAO();  
 **tranDauDoiBongDAO** = **new** TranDauDoiBongDAO();  
 }  
  
 *// Lấy danh sách thẻ phạt các đội* @Test  
 **public void** getPenaltyCardsOfAllTeams() {  
 TranDauDoiBong tddb1 = TranDauDoiBong.*builder*().id(1).tranDauId(1).doiBongId(1)  
 .maDoi(**"D1"**).tenDoi(**"MU"**).ghiChu(**"khong"**).soTheVang(2).soTheDo(3).soPhaPhamLoi(4)  
 .build();  
 TranDauDoiBong tddb2 = TranDauDoiBong.*builder*().id(2).tranDauId(1).doiBongId(2)  
 .maDoi(**"D2"**).tenDoi(**"Juve"**).ghiChu(**"good"**).soTheVang(3).soTheDo(4).soPhaPhamLoi(9)  
 .build();  
 List<TranDauDoiBong> listExpect = **new** ArrayList<>();  
 listExpect.add(tddb1);  
 listExpect.add(tddb2);  
  
 List<TranDauDoiBong> tranDauDoiBongs = **tranDauDoiBongDAO**.findPenaltyCardsOfAllTeams();  
 List<TranDauDoiBong> listActual = **new** ArrayList<>();  
 listActual.add(tranDauDoiBongs.get(0));  
 listActual.add(tranDauDoiBongs.get(1));  
  
 Assert.*assertTrue*(listActual.size() > 0);  
 Assert.*assertEquals*(listExpect, listActual);  
 }  
  
 *// Danh sách thẻ phạt trống* @Test  
 **public void** getPenaltyCardsOfAllTeams\_Empty() {  
 List<TranDauDoiBong> actual = **new** ArrayList<>();  
 Assert.*assertEquals*(0, actual.size());  
 }  
  
 *// Lấy thống kê thẻ phạt của 1 đội trong 1 trân đấu* @Test  
 **public void** getTeamPenaltyCard() {  
 ThongKe expect = ThongKe.*builder*()  
 .maDoi(**"D1"**).tenDoi(**"MU"**).tenSan(**"Hola"**)  
 .soTheVang(2).soTheDo(3).soPhaPhamLoi(4)  
 .tenVongDau(**"Vong loai 1"**).ngay(**"12/11/2021"**).gio(**"17"**)  
 .build();  
  
 List<ThongKe> actual = **thongKeDAO**.findAllPenaltyCardsByDoiBongId(1);  
  
 Assert.*assertNotNull*(actual);  
 Assert.*assertTrue*(actual.size() > 0);  
 Assert.*assertEquals*(expect, actual.get(0));  
 }  
  
 *// Danh sách thẻ phạt của 1 đội bóng chưa đá trận nào* @Test  
 **public void** getPenaltyCardsOfTeamPlayNoMatch() {  
 List<ThongKe> actual = **thongKeDAO**.findAllPenaltyCardsByDoiBongId(5);  
 **int** result = actual.size();  
 Assert.*assertEquals*(0, result);  
 }  
  
 *// Danh sách thẻ phạt của đội bóng không tồn tại* @Test  
 **public void** getPenaltyCardsOfTeamDoesntExist() {  
 List<ThongKe> actual = **thongKeDAO**.findAllPenaltyCardsByDoiBongId(999);  
 **int** result = actual.size();  
 Assert.*assertEquals*(0, result);  
 }  
  
}

## CHECKLIST

1. Tổ chức, cấu trúc dự án
2. Cài đặt, cấu hình, kết nối đến cơ sở sữ liệu
3. Cài đặt module Thống kê thẻ phạt các đội
4. Xây dựng bộ testcase cho chức năng
5. Thực hiện cài đặt kiểm thử đơn vị với chức năng đã làm

# CHƯƠNG 6. KIỂM THỬ HỆ THỐNG

## THỰC HIỆN TEST CHỨC NĂNG

Việc thực hiện test chức năng nhằm đảm bảo các chức năng của hệ thống hoạt động theo đúng yêu cầu của đặc tả.

Quy trình tiến hành các bước như sau:

* + - Bước 1: Xây dựng checklist các nội dung cần test cho từng chức năng (tuỳ chọn).
    - Bước 2: Thực hiện viết test cases
      * GUI (giao diện)
      * chức năng
      * luồng, nghiệp vụ
      * khác
    - Bước 3: Thực hiện chuẩn bị data test (nếu cần)
    - Bước 4: thực hiện test và ghi nhận kết quả pass/fail. Trường hợp fail cần mô tả rõ lỗi và trao đổi với Lập trình viên để sửa

Ta áp dụng với module Thống kê thẻ phạt chỉ với các bước 1,2,3. Riêng Bước 4, ta cần có hệ thống chạy được thì mới thực hiện được

## Bước 1. Xây dựng checklist

Bảng 6.1 Checklist test cases chức năng

|  |  |
| --- | --- |
|  | **Question** |
| Chung | Giao diện người dùng có đúng với prototype không? |
| Danh sách menu và màn hình có thống nhất không? |
| Các link html có hoạt động không? |
| Các màn hình được gọi từ button có hiển thị đúng không? |
| Màn hình KHÔNG có lỗi chính tả, cấu trúc câu, ngữ pháp? |
| Nghiệp vụ | Đã phủ hết các kịch bản valids theo đặc tả? |
| Đã phủ hết các kịch bản invalids theo đặc tả? |
| Quy trình thực hiện đã tối ưu số thao tác? |
| Quy trình thực hiện có dễ hiểu, dễ dùng? |
| Chức năng/giao diện có dễ chỉnh sửa trong trường hợp người dùng có nhu cầu thay đổi/thêm mới? |

## Bước 2. Thực hiện viết testcases

## Module Thống kê thẻ phạt

Bảng 6.2 Hướng dẫn xây dựng test case chức năng cho module Thống kê thẻ phạt

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã trường hợp kiểm thử** | **Mục đích kiểm thử** | **Các bước thực hiện** | **Kết quả mong muốn** |
|  | Chức năng Thống kê thẻ phạt | | |
| Tiền điều kiện: Ban tổ chức đăng nhập vào hệ thống | | |
| Giao diện màn hình Thống kê | | |
| TKTP\_1 | Kiểm tra tổng thể giao diện màn hình | Kiểm tra về bố cục, font chữ, chính tả, màu chữ | * Không có lỗi về chính tả, cấu trúc câu, ngữ pháp trên màn hình. * Form được bố trí hợp lý và dễ sử dụng. |
| TKTP\_2 | Kiểm tra thực hiện chức năng Thống kê thẻ phạt của các đội | Chọn button Thống kê thẻ phạt | * Hiển thị danh sách tổng số thẻ phạt của các đội bóng |
| TKTP\_3 | Kiểm tra thực hiện chức năng Xem chi tiết thẻ phạt | Chọn button Xem chi tiết | * Hiển thị danh sách thẻ phạt trong từng trận đấu của đội bóng |

## Yêu cầu xây dựng test data cho chức năng thông kê:

Do chức năng thống kê cần có dữ liệu đầy đủ trong các bảng thì mới thống kê được, do vậy việc xây dựng test data cho chức năng thống kê là quan trọng và cần thiết để phủ hết các tình huống xảy ra. Ví dụ về yêu cầu xây dựng test data cho chức năng thống kê:

|  |  |  |
| --- | --- | --- |
| **STT** | **Mục** | **Yêu cầu dữ liệu** |
| 1 | Số lượng ban tổ chức | 1 |
| 2 | Số lượng đội bóng | >=4 |
| 3 | Số lượng cầu thủ mỗi đội | >= 2 |
| 4 | Số lượng giải đấu | >= 1 |
| 5 | Số lượng sân đấu | >= 2 |
| 6 | Số lượng trận đấu | >= 2 |
| 7 | Số lượng trọng tài | >= 1 |
| 8 | Số lượng vòng đấu | >= 1 |