|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC SÀI GÒN | **KỲ THI KẾT THÚC HỌC PHẦN** |
|  | **Học phần:        Mã học phần: 841052**  **Học Kỳ 2      Năm học 2020-2021** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Họ tên sinh viên : Trần Phan Thanh Long**  **Ngày sinh**:  05/06/2000 | | | **Mã số sinh viên**  3118410248 | **Mã nhóm thi**  841052\_2002 |
| **Điểm** | | Họ tên (chữ ký) cán bộ chấm thi 1 | Họ tên (chữ ký) cán bộ chấm thi 2 | |
| *Bài tập* | *Phát vấn* |

Bài làm

# Câu 1. Giới thiệu kiến trúc n-tier. Phân biệt Tier và Layer.

Kiến trúc N-tier còn được gọi là multi-tier architecture, một phương pháp kiến trúc ứng dụng trong phát triển phần mềm.

Gọi là N-tier điều đó có nghĩa kiến trúc này có thể có 1, 2, 3 hoặc hơn số các layer phụ thuộc vào cách phân chia kiến trúc hệ thống. Tuy nhiên 3-Tier vẫn là mẫu phổ biến nhất và được định nghĩa cụ thể về trách nhiệm từng tier như sau:

# Câu 2. Giải thích tại sao phải xây dựng phần mềm theo kiến trúc n-tier?

Với kiến trúc 3-tier, ứng dụng sẽ đảm bảo các tính chất **Scalability, Data Integrity, Reusability, Reduced Distribution, Security, Improved Availability**.

# Câu 3. Giới thiệu mô hình kiến trúc 2-tier.

**2-Tier Architecture**: cũng như kiến trúc Client – Server, các giao tiếp chỉ xảy ra giữa Client – Server. Nó bảo gồm Presentation layer (UI) chạy phía client và 1 tầng làm nhiệm vụ xử lý, thực thi, lưu trữ phía Server.

Không có tầng Business Logic hay tầng trung gian nào giữa giao diện và database. Đây có lẽ là kiến trúc mà đa số các bạn “vô tình” sử dụng mà không biết khi còn là Newbie chưa có khái niệm nhiều về Architecture. Đơn giả như chúng ta xây dựng các ứng dụng web nhỏ với 1 đống code connection, query trực tiếp và DB lấy dữ liệu đổ ra view ..v.v.v..

# Câu 4. Hãy giải thích chức năng, nhiệm vụ của Presentation tier, Application tier, Data tier trong mô hình kiến trúc 3-tier?

**Tầng Presentation:** Chính là Giao diện người dùng, đây chính là phần mềm ứng dụng mà người dùng sẽ thấy và tương tác (Có thể là Website hoặc Mobile App, hoặc Window app). Khi người dùng nhập thông tin họ cần. Hành động người dùng được xử lí đi qua các tầng Logic, tầng Data.

**Tầng Logic:** Đây là tầng chứa tất cả những phương pháp xử lý, đọc & ghi dữ liệu trước khi đưa đến UI người dùng, nó những gì được cho phép trong ứng dụng của bạn.

**Tầng Data:** Tầng Data là nơi lưu trữ tất cả dữ liệu trong ứng dụng, tại tầng này bạn thực hiện các phương thức lưu trữ dữ liệu vào DB, triển khai các giải pháp bảo mật, transaction cần thiết.

# Câu 5. Trình bày sơ đồ quan hệ thực thể Entity Relationship Diagram (ER diagram).

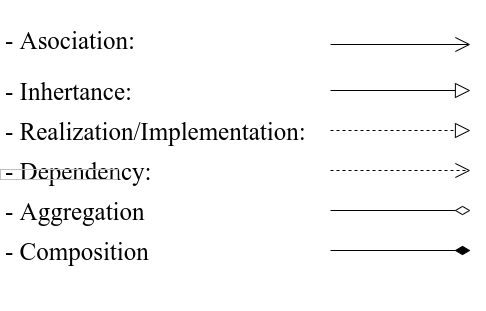
Từ những ý nghĩa này ta có thể định nghĩa được Entity Relationship Diagram là một sơ đồ hiển thị mối quan hệ của các tập thực thể được lưu trữ trong cơ sở dữ liệu. Nói cách khác, Entity Relationship Diagram giúp giải thích cấu trúc logic của cơ sở dữ liệu. Sơ đồ ER được tạo dựa trên ba khái niệm cơ bản: thực thể, thuộc tính và mối quan hệ.

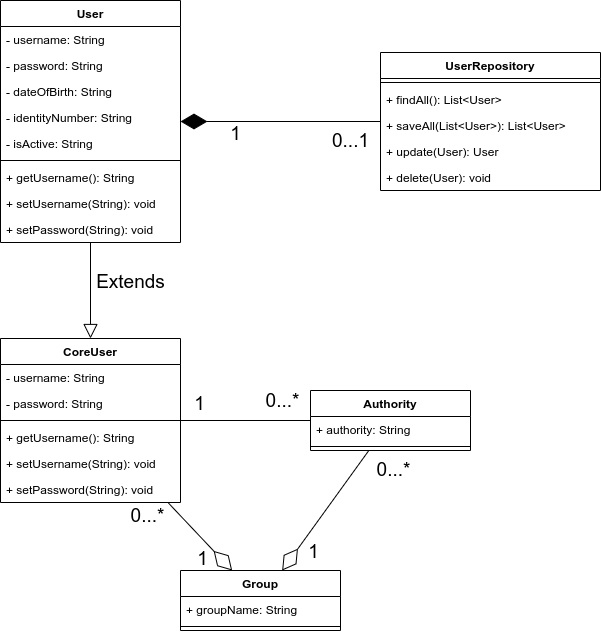
# Câu 6. Trình bày Domain Model, Class Model. Các loại quan hệ của Class model, vẽ sơ đồ và cho ví dụ minh họa.

Domain Model là một hệ thống trừu tượng mô tả các khía cạnh được lựa chọn của một phạm vi kiến thức, ảnh hưởng hoặc hoạt động (một domain). Sau đó, model có thể được sử dụng để giải quyết các vấn đề liên quan đến domain đó. Domain model là sự trình bày các khái niệm trong thế giới thực có ý nghĩa liên quan đến doamin cần được mô hình hóa trong phần mềm. Các khái niệm bao gồm business data và các ràng buộc liên quan đến data đó. Domain model thường sử dụng ngôn ngữ tự nhiên.

Mô hình lớp xác định cấu trúc của toàn bộ hệ thống bằng cách xác định cấu trúc tĩnh của các đối tượng trong hệ thống đó. Một class model xác định các thuộc tính và hoạt động cho các đối tượng của mỗi lớp và cả mối quan hệ giữa các đối tượng trong hệ thống.

Các loại quan hệ class model:





# Câu 7. Phân biệt 2 đối tượng Data Tranfer Object (DTO) và Entity.

Entity có thể là một phần của business domain. Do đó, chúng có thể triển khai hành vi (phương thức) và được áp dụng cho các trường hợp sử dụng khác nhau trong domain.

DTO chỉ được sử dụng để chuyển dữ liệu từ quy trình hoặc ngữ cảnh này sang quy trình hoặc ngữ cảnh khác. Do đó, chúng không có hành vi (phương thức) - ngoại trừ các chức năng lưu trữ và truy xuất rất cơ bản và thường được tiêu chuẩn hóa.

# Câu 8. Hãy giải thích chức năng, nhiệm vụ của từng lớp Model, Controller, View. Trong ứng dụng có thể gộp các lớp nào lại với nhau?

Model có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là một cầu nối giữa thành phần bên dưới là View và Controller. Model thể hiện, giao tiếp với một database hoặc bất kỳ nguồn dữ liệu nào khác. Model thể hiện rõ các thao tác với database như cho phép xem, truy xuất, xử lý dữ liệu.

View: Đây là phần giao diện (theme) dành cho người sử dụng. Nơi mà người dùng có thể lấy được thông tin dữ liệu của chương trình thông qua các thao tác truy vấn như tìm kiếm hoặc sử dụng thông qua các browser.

Thông thường, các ứng dụng web sử dụng View như một phần của hệ thống, nơi các thành phần [HTML](https://monamedia.co/html-va-css-la-gi/) được tạo ra. Bên cạnh đó, View cũng có chức năng ghi nhận hoạt động của người dùng để tương tác với Controller. Tuy nhiên, View không có mối quan hệ trực tiếp với Controller, cũng không được lấy dữ liệu từ Controller mà chỉ hiển thị yêu cầu chuyển cho Controller mà thôi.

Controller: Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua view. Từ đó, đưa ra dữ liệu phù hợp với người dùng. Bên cạnh đó, Controller còn có chức năng kết nối với model. Chức năng chính của controller là xử lý business logic.

Trong ứng dụng, đôi khi không có lớp đại diện cho database, mà dùng một lớp duy nhất để truy xuất, cập nhật database và xử lý luôn business logic, tức lớp controller và model được gộp lại với nhau.

# Câu 9. Giải thích về Route trong mô hình MVC.

**Routes** được sử dụng định tuyến người dùng theo đúng **Urls**. Ở đây, mọi **Request** từ phía người dùng đều phải qua **Route**, dữ liệu được gửi xuống **Controller**để xử lý, cần dữ liệu sẽ lấy từ **Model**lên hoặc cập nhật dữ liệu xuống **Model**, kết quả gửi ra **View**cho người sử dụng.

# Câu 10. So sánh mô hình 3-layer và mô hình MVC.

**Điểm giống nhau:**

- Cả hai đều là mô hình kiến trúc áp dụng trong lập trình được tách thành 3 thành phần.

- Tách biệt sự phụ thuộc giữa ngôn ngữ lập trình, môi trường phát triển, xử lý logic.

- Tương đồng về chức năng của thành phần trình bày giao diện.

Điểm Khác nhau:

|  |  |
| --- | --- |
| **3-Layer** | **MVC** |
| Flow đi tuần tự từ GUI -> BLL -> DAO -> BLL -> GUI | Controller là tầng điểu phối của các ứng dụng: nhận request từ client, phân tích request để gọi Model đến để lấy dữ liệu và tầng View tổ chức hiển thị dữ liệu |
| DAO chỉ dùng để giao tiếp với database. | Model đôi khi giữ chức năng xử lý business và data access. |
| Thường được sử dụng trong các desktop app. | Thường được sử dụng trong webapp. |

# Câu 11. Entity Framework là gì, làm việc như thế nào?

Entity Framework là một ORM framework mã nguồn mở cho các ứng dụng .NET được Microsoft hỗ trợ. Nó cho phép các nhà phát triển làm việc với dữ liệu bằng cách sử dụng các đối tượng của các lớp cụ thể trong domain mà không cần tập trung vào các bảng và cột cơ sở dữ liệu bên dưới nơi dữ liệu. Với Entity Framework, các nhà phát triển có thể làm việc ở mức độ trừu tượng cao hơn khi họ xử lý dữ liệu, đồng thời có thể tạo và duy trì các ứng dụng hướng dữ liệu với ít mã hơn so với các ứng dụng truyền thống.

# Câu 12. Các cú pháp truy vấn dữ liệu với LINQ to SQL.

- Mệnh đề from:

var result = **from** product **in** products **select** product;

- Mệnh đề select:

var result = **from** product **in** products **select** product;

- Lọc dữ liệu Where:

var result = **from** product **in** products **where** product.price >= 10000 **select** product;

- Sắp xếp kết quả OrderBy:

var result = **from** product **in** products **orderby** product.id **descending select** product;

- GroupBy:

var result = **from** product **in** products **where** product.price <= 1000 && product.price >=2000 **group** product **by** product.price;

- Join:

**from** product **in** products **join** category **in** categories **on** product.category **equals** category.id **where** category.id == 1 **select** product;

# Câu 13. Các cú pháp truy vấn dữ liệu với LINQ to Entity.

- FirstOrDefault:

using(var ctx = new ProductsDBEntities())

{

var result = (**from** product **in** ctx.Products **select** product).FirstOrDefault<Product>();

}

- Single/SingleOrDefault:

using(var ctx = new ProductsDBEntities())

{

var result = ( **from** product **in** ctx.Products **select** product ).SingleOrDefault<Product>();

}

- Lọc dữ liệu ToList:

using(var ctx = new ProductsDBEntities())

{

var result = (**from** product **in** ctx.Products **select** product).ToList<Product>();

}

- Sắp xếp kết quả OrderBy:

using(var ctx = new ProductsDBEntities())

{

var result = ( **from** product **in** ctx.Products **orderby** product.price **ascending** **select** product ).ToList<Product>();

}

- GroupBy:

using(var ctx = new ProductsDBEntities())

{

var result = ( **from** product **in** ctx.Products **group** product **by** product.color **into** productByColor **select** productByColor );

}