

1. Mô tả các chức năng bằng ngôn ngữ tự nhiên

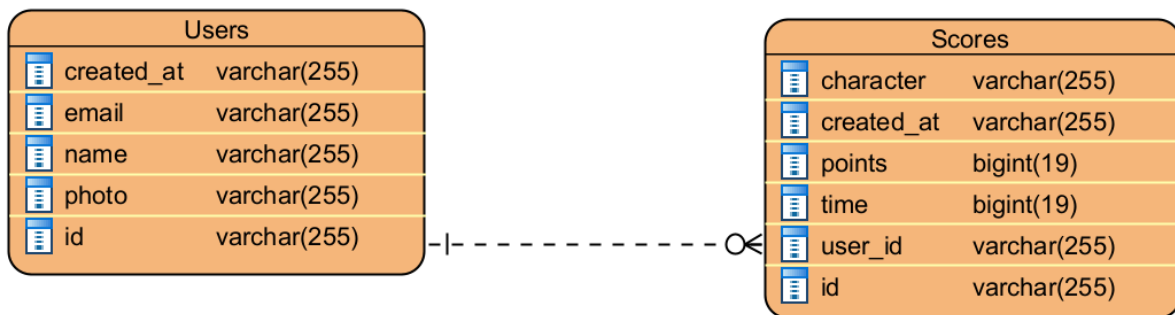
- Chọn nhân vật: ở màn hình đầu tiên người dùng sẽ thấy một hoạt ảnh nhỏ của nhân vật mặc định ở góc trên bên phải màn hình, khi nhấn vào hoạt ảnh sẽ hiển thị ra màn hình chọn nhân vật. Màn hình nhân vật sẽ hiển thị nhân tên và hoạt ảnh lớn của nhân vật đang được chọn, khi bấm nút chuyển nhân vật sẽ chuyển tên và hoạt ảnh đang hiển thị sang nhân vật khác (có hai nút chuyển phía trước và phía sau). Bấm nút chọn nhân vật, sẽ cập nhật nhân vật được chọn và chuyển về màn hình chính, màn hình chính sẽ hiển thị hoạt ảnh nhỏ của nhân vật được chọn ở góc trên bên phải màn hình.
- Lưu lịch sử chơi, hiển thị kết quả chơi: sau khi người chơi kết thúc trò chơi (bị hết mạng hoặc vượt qua tất cả các màn chơi), kết quả chơi sẽ được lưu lại vào thiết bị (nếu có kết nối mạng kết quả cũng sẽ được lưu lại vào cơ sở dữ liệu trên server), màn hình sẽ chuyển hướng đến màn hình kết quả và hiển thị tên nhân vật đã chọn, điểm số đạt được, thời gian chơi (tính bằng giây) và thời gian kết thúc trò chơi.
- Cấu hình âm nhạc game: khi ở màn hình chính, người chơi sẽ nhìn thấy biểu tượng loa ở góc trên bên phải màn hình, mặc định là âm nhạc được bật, khi chạm vào biểu tượng loa sẽ tắt/bật nhạc. Khi nhạc được tắt/bật, người dùng chuyển màn hình khác cũng đều nghe được nhạc đang tắt/bật, ngay cả trong thế giới game. Trong thế giới game cũng có thể bật/tắt nhạc bằng cách chạm vào biểu tượng loa.
- Xem lịch sử chơi: ở màn hình chính, người dùng bấm chọn nút lịch sử chơi, màn hình lịch sử chơi sẽ hiển thị lịch sử chơi, lịch sử chơi sẽ được sắp xếp từ mới nhất đến cũ nhất, mặc định lịch sử chơi sẽ được đọc từ cơ sở dữ liệu trên thiết bị, để tránh việc khi không có kết nối mạng vẫn có thể xem được lịch sử chơi, lịch sử chơi được đồng bộ 2 chiều giữa server với thiết bị khi đã đăng nhập tài khoản và có kết nối mạng.
- Đăng nhập: ở màn hình chính người dùng sẽ thấy ảnh đại diện mặc định và tên guest ở góc trên bên tay trái màn hình, khi bấm vào đây màn hình hiển thị 2 tùy chọn đăng nhập hoặc đăng ký. Người dùng chọn đăng nhập, màn hình đăng nhập sẽ hiện lên yêu cầu người dùng nhập email và mật khẩu của tài khoản đã đăng ký để đăng nhập, khi đăng nhập thành công màn hình đăng nhập sẽ chuyển hướng về màn hình chính đồng thời hiển thị tên, ảnh đại diện, id của tài khoản và tài lịch sử chơi của tài khoản đó về thiết bị.

- Đăng ký: ở màn hình chính người dùng sẽ thấy ảnh đại diện mặc định và tên guest ở góc trên bên tay trái màn hình, khi bấm vào đây màn hình hiển thị 2 tùy chọn đăng nhập hoặc đăng ký. Người dùng chọn đăng ký, màn hình đăng ký hiện lên yêu cầu người dùng nhập các thông tin cần thiết để đăng ký tài khoản (email, tên, mật khẩu, mật khẩu nhập lại) và người dùng có thể chọn ảnh đại diện cho tài khoản của mình hoặc không. Khi đăng ký thành công màn hình sẽ chuyển hướng đến màn hình chính đồng thời hiển thị id, tên, ảnh đại diện của tài khoản và xoá toàn bộ lịch sử chơi hiện tại trên thiết bị.
- Xem thông tin tài khoản: ở màn hình chính, người dùng sẽ thấy ảnh đại diện, id và tên của tài khoản, khi bấm vào sẽ chuyển hướng đến màn hình hiển thị thông tin tài khoản bao gồm: ảnh đại diện, id tài khoản, tên tài khoản, email tài khoản và ngày tạo tài khoản.
- Cập nhật thông tin tài khoản: ở màn hình xem thông tin cá nhân, người dùng sẽ thấy biểu tượng hình bút chì ngay bên phải id của tài khoản, khi bấm vào màn hình thay đổi thông tin sẽ hiện ra cho phép người dùng cập nhật tên và ảnh đại diện cho tài khoản của họ. Mặc định màn hình sẽ hiển thị ảnh đại diện hiện tại và tên hiện tại của tài khoản. Lúc này, người dùng bấm vào ảnh đại diện của mình sẽ được chọn ảnh trên thiết bị để thay đổi, người dùng bấm vào textfield chứa tên tài khoản của mình để đổi tên. Khi hoàn tất việc thay đổi thông tin, bấm lưu thay đổi để cập nhật thay đổi, cập nhật thay đổi thành công màn hình sẽ chuyển hướng đến màn hình xem thông tin tài khoản và hiển thị thông tin đã được cập nhật.
- Đăng xuất: ở màn hình xem thông tin tài khoản người dùng sẽ thấy nút đăng xuất ở góc trên bên phải màn hình, khi bấm vào sẽ đồng bộ lịch sử chơi ở thiết bị lên server sau đó xoá toàn bộ lịch sử chơi khỏi thiết bị và đăng xuất tài khoản khỏi thiết bị.
- Đồng bộ dữ liệu lịch sử chơi: Chức năng này là một dịch vụ chạy nền, khi ứng dụng được mở sẽ lắng nghe trạng thái mạng, khi có kết nối sẽ đồng bộ lịch sử chơi từ thiết bị lên server và từ server về thiết bị. Chức năng này hoạt động kể cả khi mở ứng dụng mà không có kết nối mạng, dịch vụ vẫn sẽ lắng nghe trạng thái mạng, khi đột ngột có kết nối mạng sẽ thực hiện đồng bộ như trên.
- Xem bảng xếp hạng: ở màn hình chính, người dùng sẽ nhìn thấy biểu tượng xếp hạng ở góc trên bên trái màn hình, khi bấm vào biểu tượng, sẽ thực hiện

tải dữ liệu sau đó thực hiện sắp xếp và xếp hạng người chơi, cuối cùng sẽ hiển thị 5 người chơi có kết quả cao nhất hiển thị từ cao đến thấp. Xếp hạng được tính theo điểm, thời gian chơi, thời gian mà kết quả đó được tạo ra. Nếu ai có điểm cao hơn sẽ xếp cao hơn, nếu số điểm của hai người là bằng nhau thì người có thời gian chơi ít hơn sẽ xếp cao hơn, nếu điểm và thời gian chơi của hai người là bằng nhau thì người đạt được kết quả đó sớm hơn sẽ được xếp cao hơn.

2. Thiết kế cơ sở dữ liệu

- Thiết kế:



- Mô tả bảng Users:

- Mục đích: lưu trữ thông tin tài khoản sau khi đăng ký thành công.
- Cột id: làm khoá chính của bảng, có kiểu varchar, được tạo bằng uuid.
- Cột email: đại diện cho email tài khoản của người dùng, có kiểu là varchar.
- Cột name: đại diện cho tên của người dùng, có kiểu là varchar.
- Cột photo: đại diện cho ảnh đại diện của người dùng đã được mã hoá dưới dạng base64, có kiểu là varchar.
- Cột created_at: đại diện cho ngày tạo tài khoản của người dùng, có kiểu là varchar.

- Mô tả bảng Scores:

- Mục đích: lưu lại điểm và các thông tin liên quan sau mỗi lần chơi của người dùng.
- Cột id: làm khoá chính của bảng, có kiểu varchar, được tạo bằng uuid.
- Cột character: đại diện cho nhân vật mà người chơi đã chọn trong lần chơi đó, có kiểu varchar.

- Cột `created_at`: đại diện cho thời gian mà bản ghi được tạo ra trên thiết bị chơi, có kiểu `varchar`.
- Cột `points`: đại diện cho số điểm mà người dùng đạt được trong lần chơi đó, có kiểu `bigint`.
- Cột `time`: đại diện cho thời gian mà người dùng chơi trong lần chơi đó (tính bằng giây), có kiểu `bigint`.
- Cột `user_id`: đại diện cho id của tài khoản người dùng, có kiểu `varchar`.
- Mỗi quan hệ: `Users – Scores` là mối quan hệ một – nhiều. Một tài khoản có lịch chơi của tài khoản đó với mỗi bản ghi được gọi là `score`.
- Sử dụng `Firestore` để triển khai cơ sở dữ liệu phía server dưới dạng `collection` có các bản ghi là `document`.
- Collection `users`:

Trường `created_at` có kiểu dữ liệu là `string`

Trường `email` có kiểu dữ liệu là `string`

Trường `name` có kiểu dữ liệu là `string`

Trường `photo` có kiểu dữ liệu là `string`

Bản ghi mẫu

<div>+ Start collection</div> <div>scores</div> <div>users ></div>	<div>+ Add document</div> <div>XcbK21GgueYU4wqSh7z2M0HP3ty1 ></div> <div>g1B1A7Fuc8Pu9z3o3VBfQs7g1C62</div> <div>n0amTxei6PenJ7pn0b0qxs95hN22</div> <div>x0BKnTHQyqVH4sS49DoVih8q89J3</div>	<div>+ Start collection</div> <div>+ Add field</div> <div>created_at: "2024-11-04 01:08:16.374365"</div> <div>email: "taso@gmail.com"</div> <div>name: "Thẩm Ấu Sở"</div> <div>photo: "iVBORw0KGgoAAAANSUgAAACKAAAMgCAYAAABI65kSAAAYr</div>
---	--	---

- Collection `scores`:

Trường `character` có kiểu dữ liệu là `string`

Trường created_at có kiểu dữ liệu là string

Trường points có kiểu dữ liệu là number

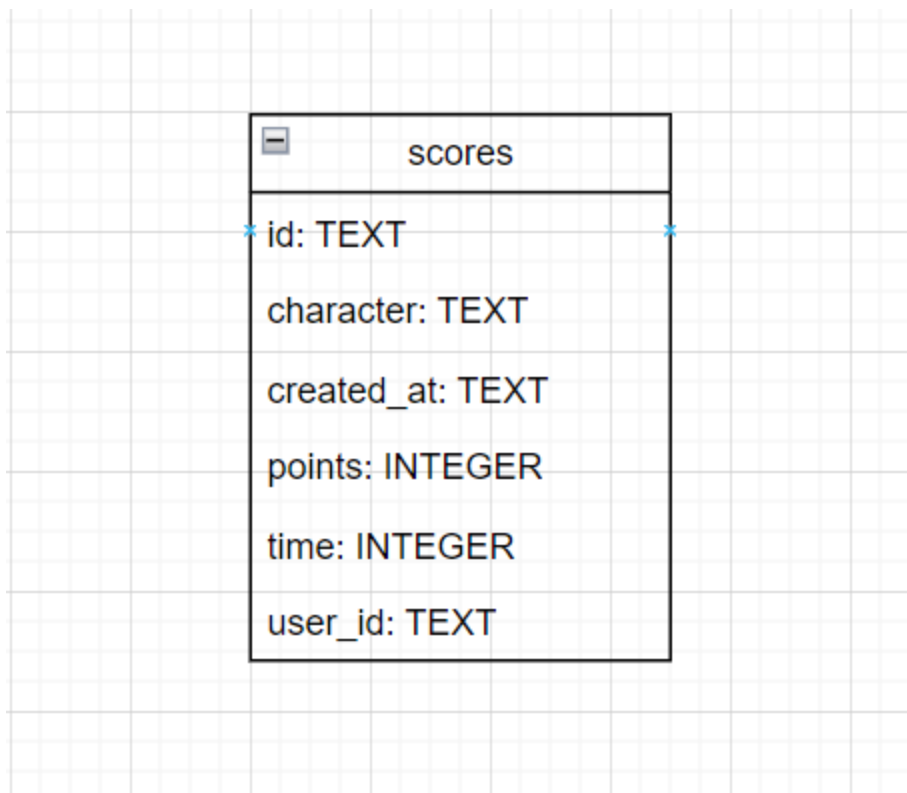
Trường time có kiểu dữ liệu là number

Trường user_id: có kiểu dữ liệu là string

Bản ghi mẫu

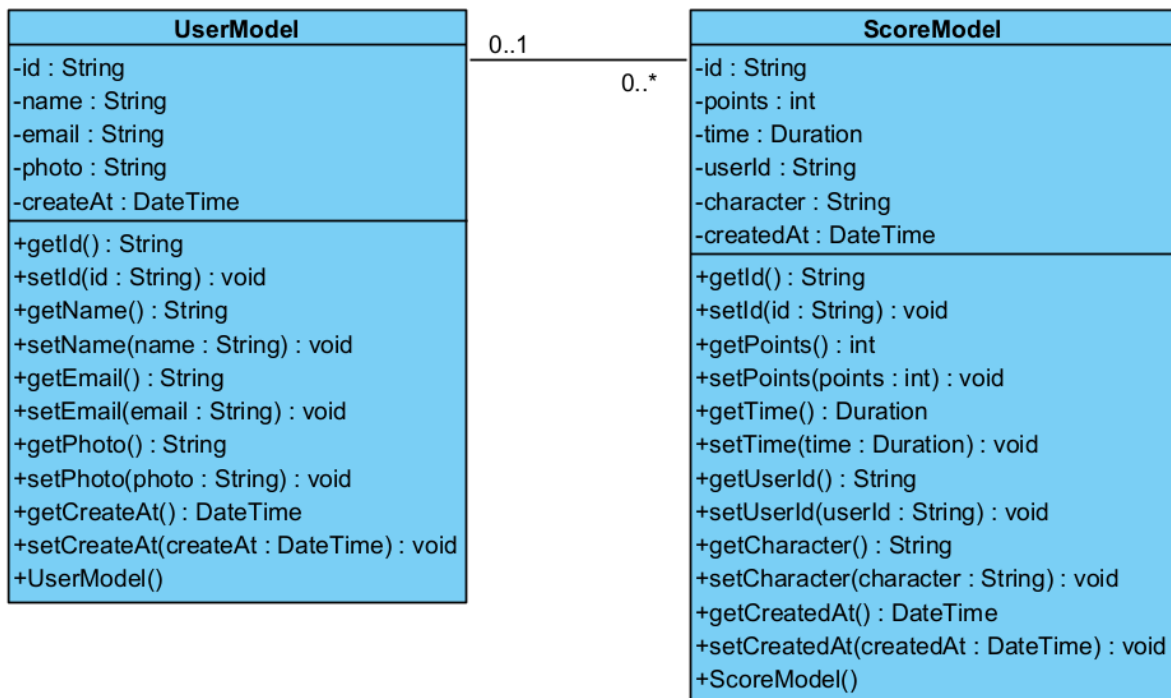
<div>+ Start collection</div> <div>scores ></div> <div>users</div>	<div>+ Add document</div> <div>541620a2-0ca1-4238-9813-9501...</div> <div>69a27976-82d9-4a72-b930-b81c... ></div> <div>74282997-56de-479d-b4c3-c62b...</div> <div>7eae5d23-ed12-4361-9a8c-8600...</div> <div>8904e6dc-8cd7-41bf-97e7-2eed...</div> <div>943f3626-bc8e-42b6-ba58-f4b3...</div> <div>9dc19143-9a45-4f49-942c-05ca...</div> <div>b017c0ba-cce4-40cf-83dc-4f11...</div> <div>d8cffacf-b7dd-48dc-98c0-2594...</div> <div>e0f47816-8b76-469c-b75f-dc92...</div>	<div>+ Start collection</div> <div>+ Add field</div> <div>character: "Mask Dude"</div> <div>created_at: "2024-11-03 18:09:27.896330"</div> <div>points: 0</div> <div>time: 9</div> <div>user_id: "n0amTxei6PenJ7pn0b0qxs95hN22"</div>
---	--	---

- Tạo luật cho firebase để bảo vệ và thiết lập quyền truy cập cho các collections:
- Với collection users sẽ cho phép đọc, không cho phép xóa, cho phép tạo nếu id tài khoản đã đăng ký là id của bản ghi, cho phép sửa nếu đã đăng nhập và id của tài khoản giống với id của bản ghi.
- Với collection scores sẽ cho phép đọc, không cho phép xóa và sửa, cho phép tạo nếu người dùng đã đăng nhập và user_id của bản ghi giống với id của tài khoản.
- Ngoài ra, không cho phép tạo mới hay đọc các collection khác.
- Sử dụng sqflite để lưu trữ dữ liệu scores trên thiết bị:



- Cột character có kiểu dữ liệu là TEXT, đại diện cho nhân vật đã sử dụng để chơi trong lần chơi đó.
- Cột created_at có kiểu dữ liệu là TEXT, đại diện cho thời gian mà bản ghi được tạo.
- Cột id có kiểu dữ liệu là TEXT, đại diện cho id của bản ghi.
- Cột points có kiểu dữ liệu là INTEGER, đại diện cho điểm số đã đạt được trong lần chơi đó.
- Cột time có kiểu dữ liệu là INTEGER, đại diện cho thời gian đã chơi trong lần chơi đó, tính bằng giây.
- Cột user_id có kiểu dữ liệu là TEXT, đại diện cho id của tài khoản đã chơi trong lần chơi đó, nếu chơi dưới dạng khách, sẽ được lưu giá trị chuỗi là "null".

3. Thiết kế lớp thực thể

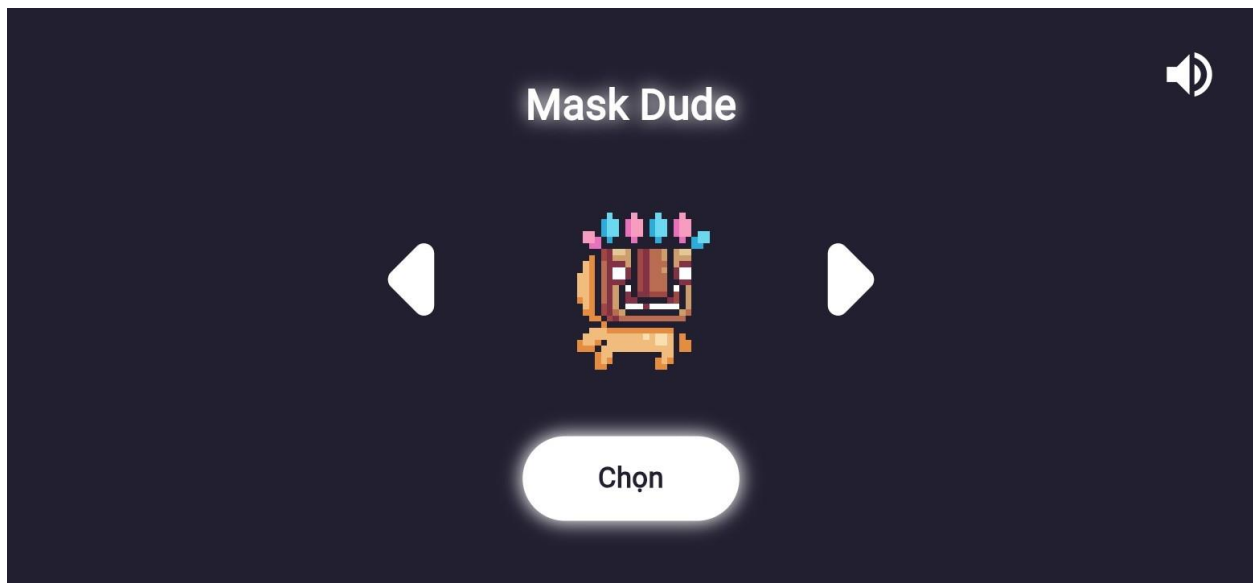


- Thực thể UserModel có các trường:
 - id: là id của tài khoản và bản ghi thông tin người dùng, có kiểu dữ liệu là String.
 - name: là tên của người dùng, có kiểu dữ liệu là String.
 - email: là email đã dùng để đăng ký tài khoản, có kiểu dữ liệu là String.
 - photo: là ảnh đại diện của người dùng được mã hoá dưới dạng base64, có kiểu dữ liệu là String.
 - createdAt: là thời gian mà người dùng đã đăng ký tài khoản, đã được chuyển dưới dạng DateTime.
- Thực thể ScoreModel có các trường:
 - id: là id của bản ghi score, có kiểu dữ liệu là String, được tạo bằng uuid.
 - points: là điểm đã đạt được trong lần chơi, có kiểu dữ liệu là int.
 - time: là thời gian chơi trong một lần chơi, được để dưới dạng Duration.
 - userId: là id của tài khoản người dùng, có kiểu dữ liệu String, nếu chơi ở chế độ khách, thực thể tạo ra sẽ có userId là chuỗi “null”.
 - character: là nhân vật mà người dùng để chọn trong lần chơi, có kiểu dữ liệu là String.

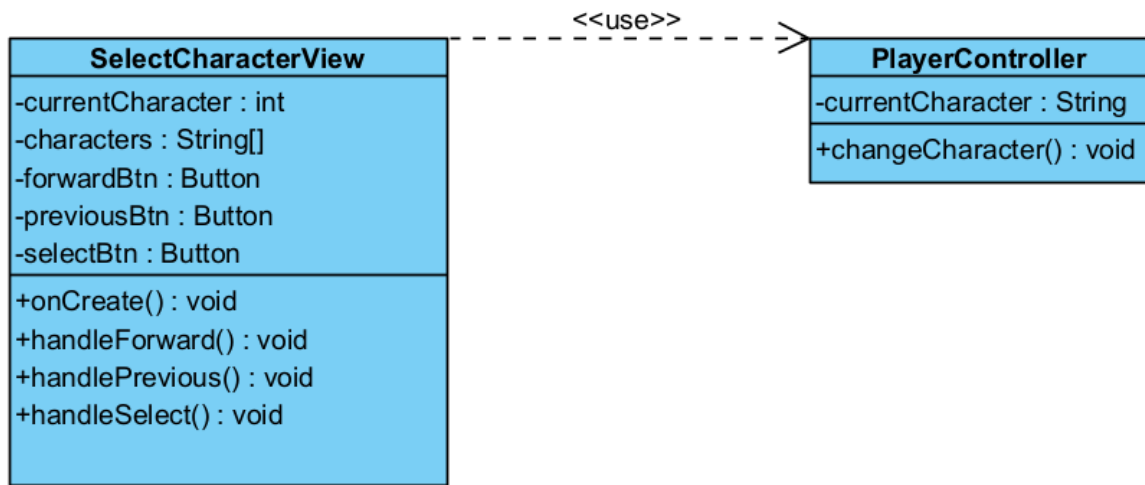
- `createdAt`: là thời gian mà thực thể được tạo (khi trò chơi kết thúc), được để dưới dạng `DateTime`.
- Một `UserModel` có thể không có hoặc có nhiều `ScoreModel`, Một `ScoreModel` có thể không thuộc về (khi người dùng chơi dưới dạng khách) hoặc thuộc về một `UserModel`.

4. Các chức năng

- Chức năng chọn nhân vật:
 - Giao diện chức năng:



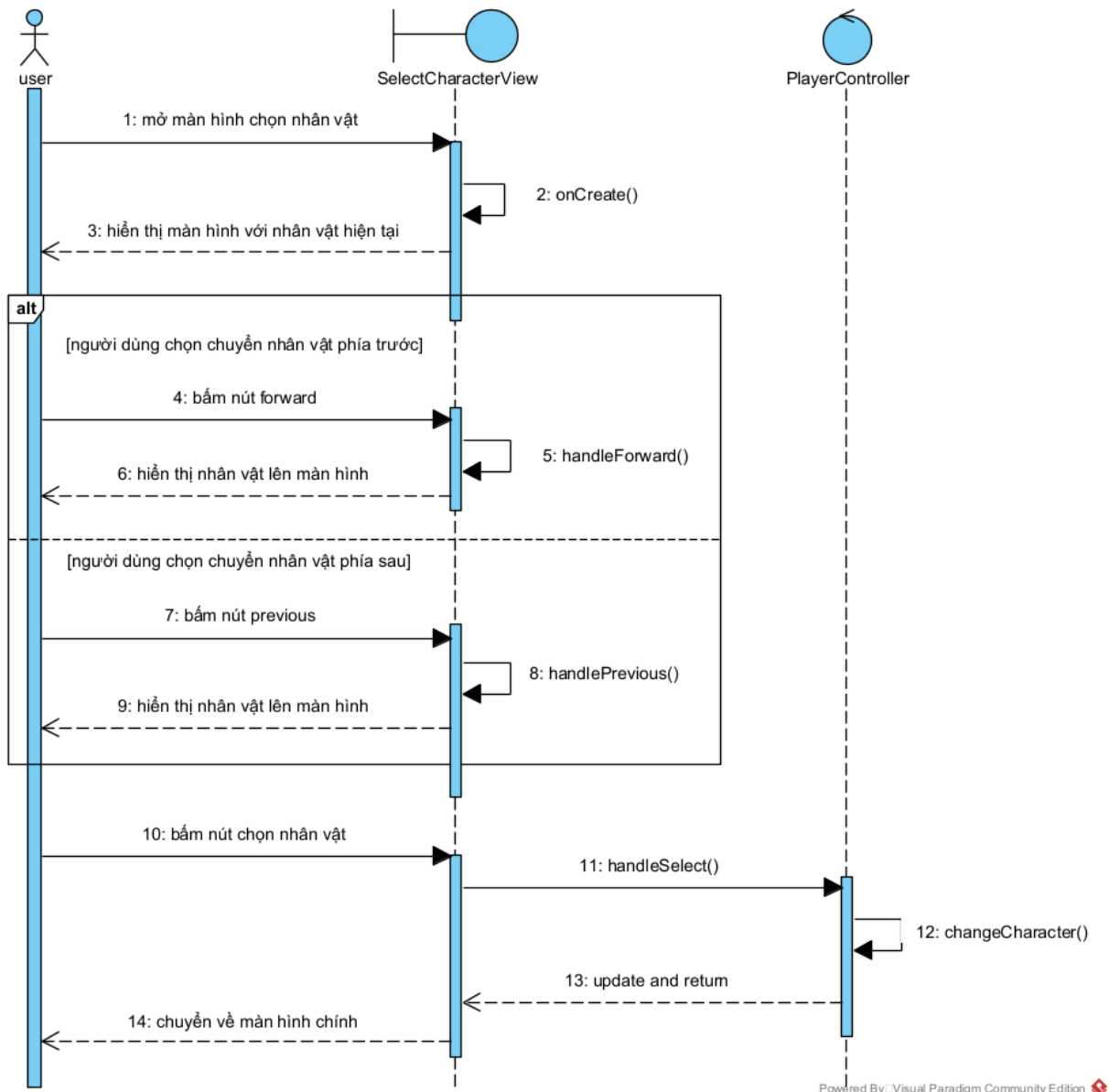
- Biểu đồ lớp chi tiết:



Lớp **SelectCharacterView** có nhiệm vụ hiển thị giao diện và bắt các sự kiện trên màn hình của người dùng như chuyển nhân vật, chọn nhân vật, hiển thị nhân vật hiện tại.

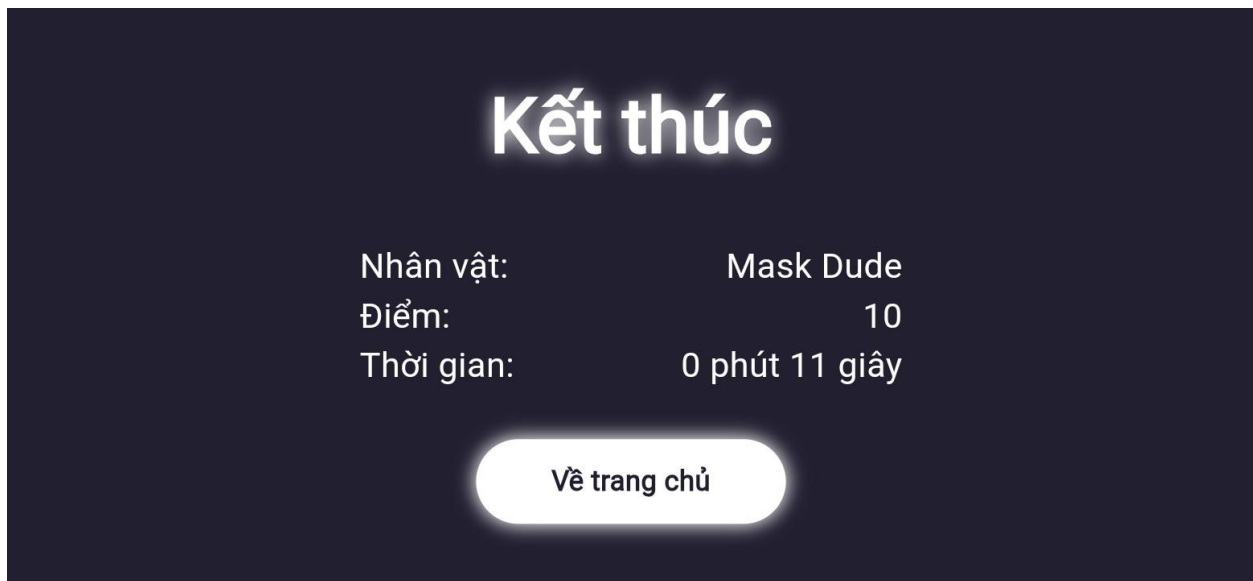
Lớp **PlayerController** có nhiệm vụ xử lý việc chọn nhân vật và lưu trữ thông tin nhân vật được chọn giữa nhiều màn khác nhau.

- Biểu đồ tuần tự chức năng:

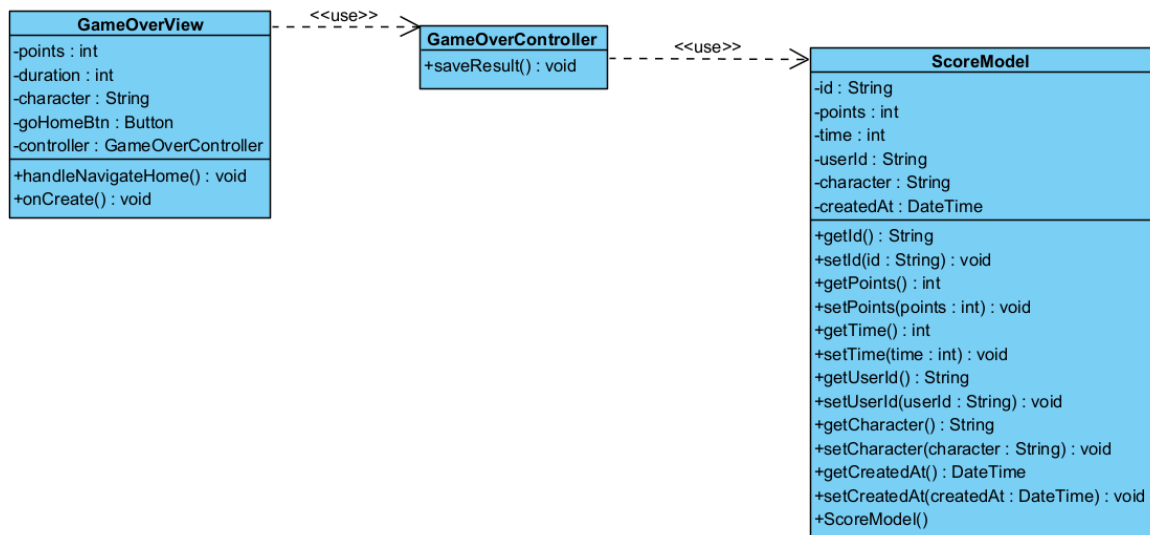


- 1) Người dùng mở màn hình chọn nhân vật.
- 2) Lớp **SelectCharacterView** khởi tạo giao diện màn hình.
- 3) Hiển thị giao diện lên màn hình thiết bị người dùng.
- 4) Người dùng bấm nút **forward** để thay đổi nhân vật đang hiển thị trên màn hình.
- 5) Lớp **SelectCharacterView** xử lý sự kiện chuyển nhân vật được hiển thị.
- 6) Hiển thị hình ảnh nhân vật lên màn hình.
- 7) Người dùng bấm nút **previous** để thay đổi nhân vật đang hiển thị trên màn hình.

- 8) Lớp SelectCharacterView xử lý sự kiện chuyển nhân vật được hiển thị.
 - 9) Hiển thị hình ảnh nhân vật lên màn hình.
 - 10) Người dùng bấm nút chọn nhân vật.
 - 11) Lớp SelectCharacterView nhận sự kiện và gọi hàm handleSelect() .
 - 12) Lớp PlayerController gọi hàm changePlayer() xử lý việc chọn nhân vật.
 - 13) Cập nhật thông tin nhân vật được chọn.
 - 14) Chuyển về màn hình chính.
- Chức năng lưu lịch sử chơi, hiển thị kết quả chơi:
 - Giao diện chức năng:



- Biểu đồ lớp chi tiết:

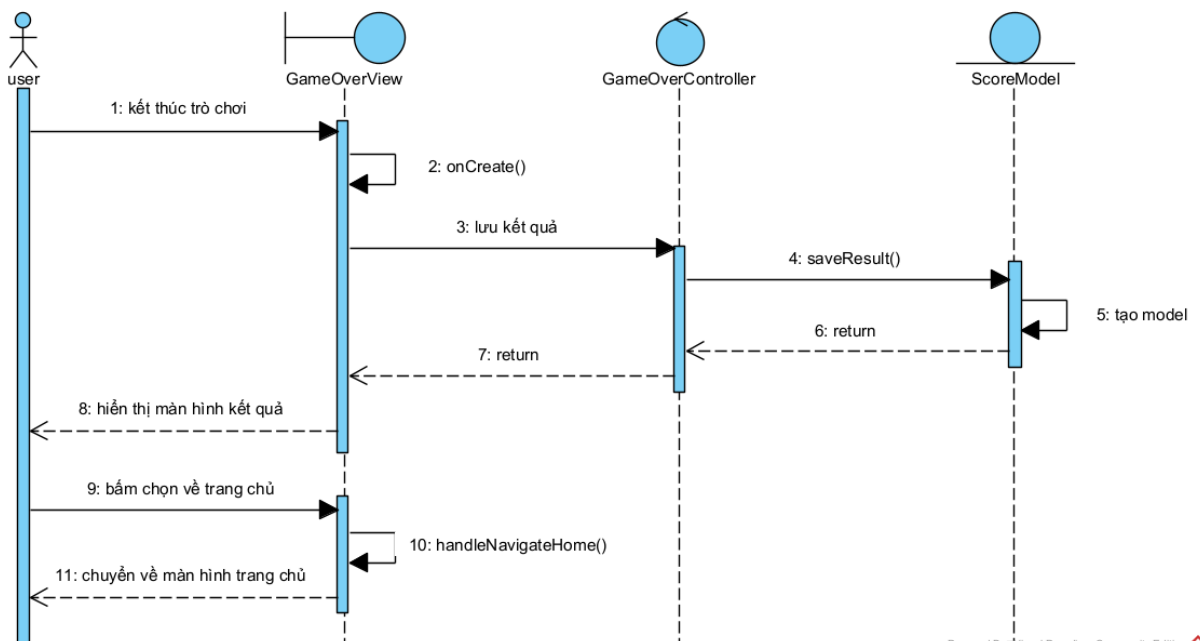


Lớp GameOverView có nhiệm vụ hiển thị giao diện kết quả trò chơi lên màn hình và bắt sự kiện chuyển về trang chủ của người dùng.

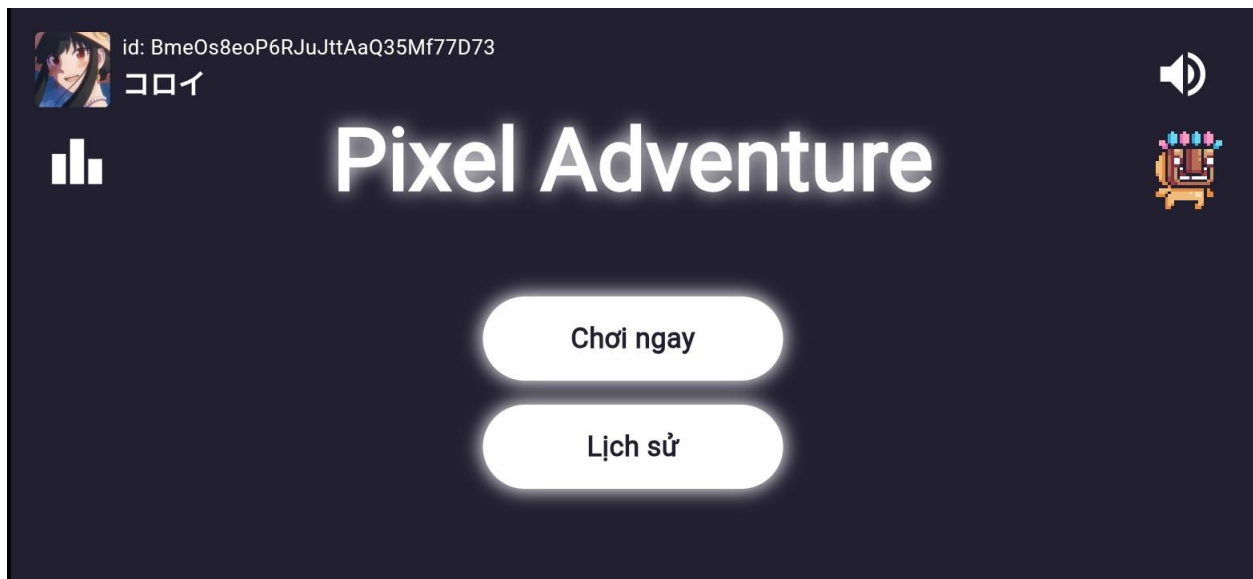
Lớp GameOverController có nhiệm vụ lưu lại kết quả chơi của người dùng.

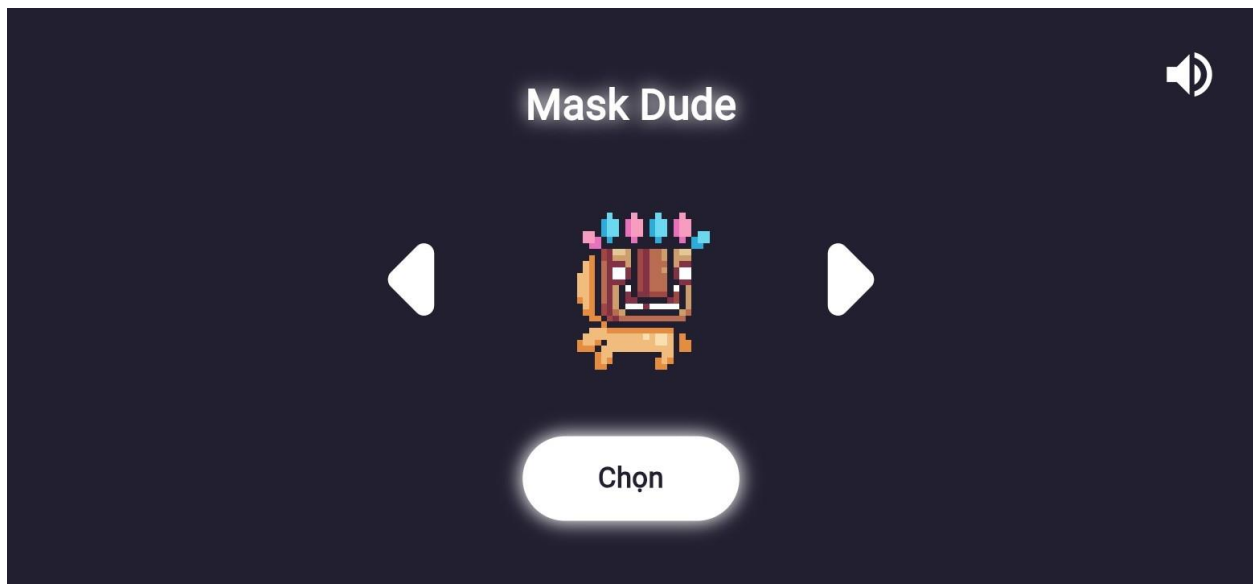
Lớp ScoreModel là lớp thực thể cho dữ liệu điểm của 1 lần chơi.

- Biểu đồ tuần tự:

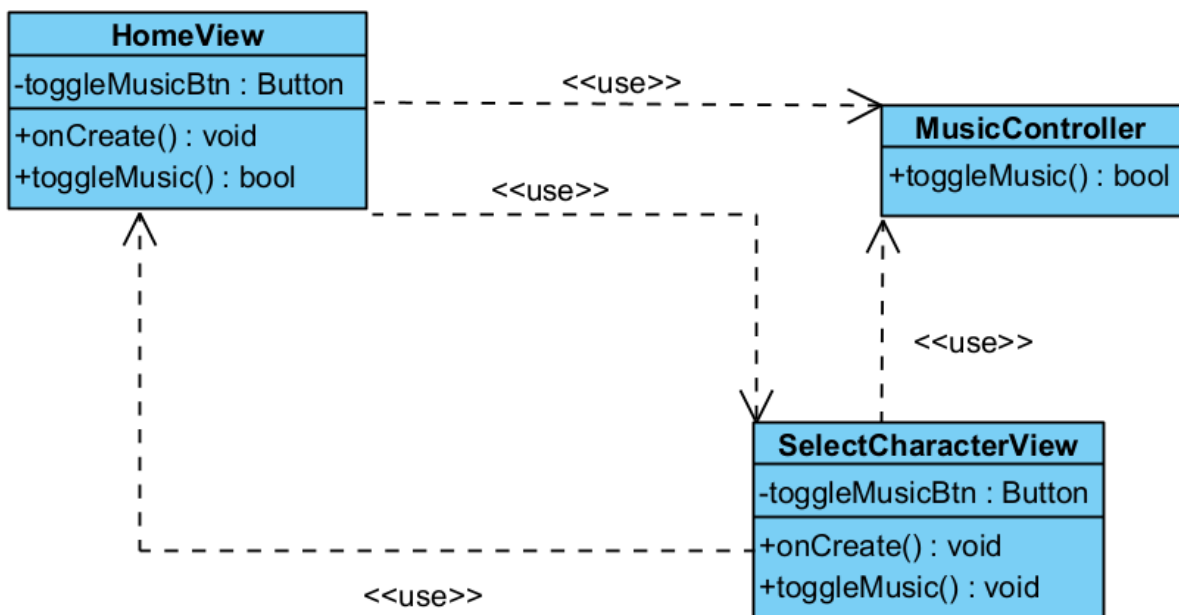


- 1) Người dùng kết thúc trò chơi.
 - 2) Lớp GameOverView gọi hàm onCreate() để khởi tạo giao diện màn hình.
 - 3) Sử dụng lớp GameOverController để lưu lại kết quả chơi.
 - 4) Lớp GameOverController gọi hàm saveResult() để lưu lại kết quả chơi.
 - 5) Tạo ScoreModel sau khi lưu thành công.
 - 6) Trả về ScoreModel đã được tạo.
 - 7) Trả về ScoreModel cho GameOverView.
 - 8) Hiển thị thông tin điểm lên màn hình.
 - 9) Người dùng bấm chọn chuyển về trang chủ
 - 10) GameOverView bắt sự kiện chuyển về trang chủ và điều hướng về trang chủ.
 - 11) Chuyển về màn hình trang chủ.
- Chức năng cấu hình âm nhạc game:
 - Giao diện chức năng: có thể bật/tắt âm nhạc trên phạm vi toàn cục ứng dụng ở màn hình chính, màn chọn nhân vật và trong thế giới game.





- Biểu đồ lớp chi tiết:

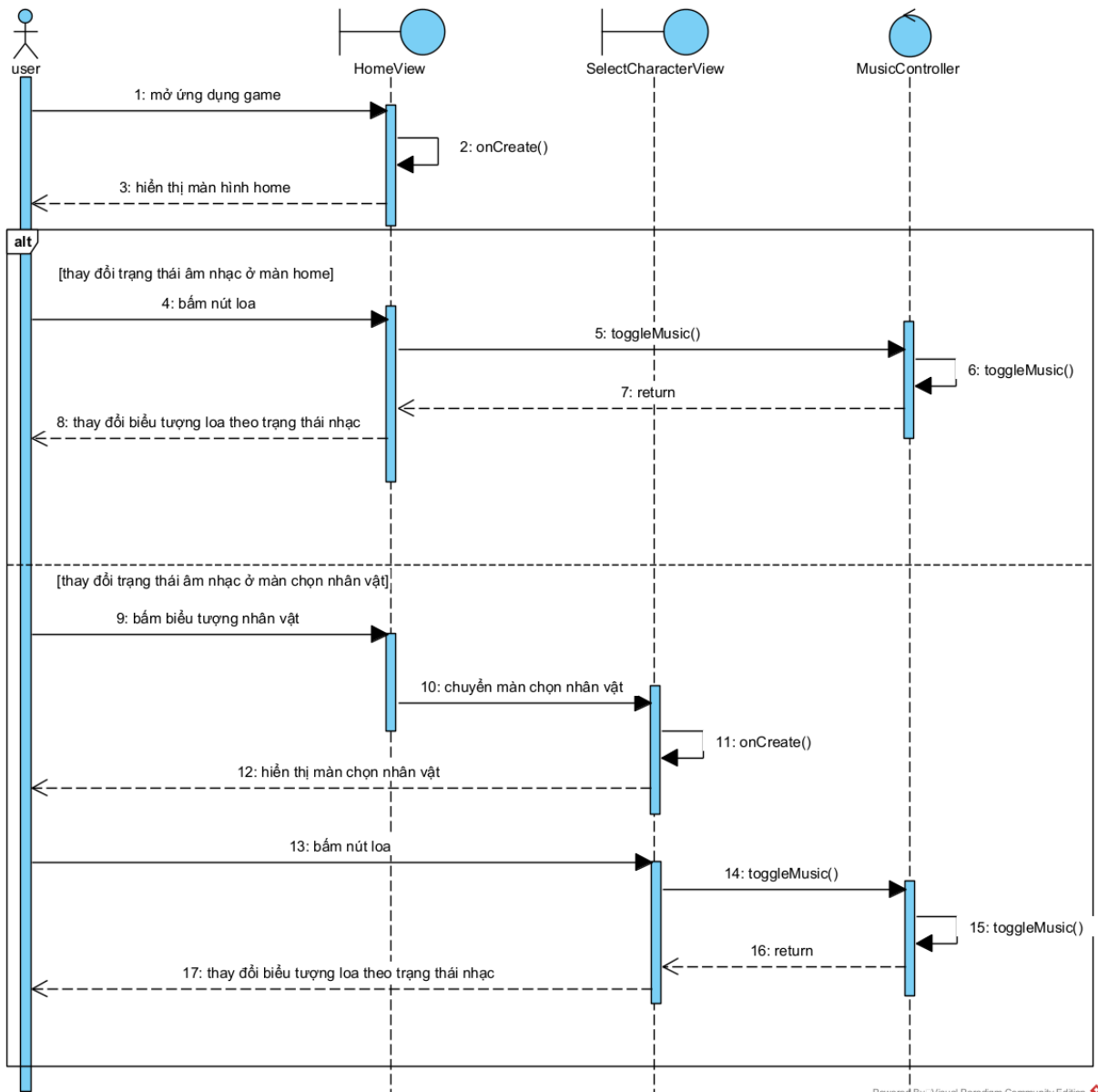


Lớp HomeView có nhiệm vụ hiển thị giao diện trang chủ và bắt các sự kiện trên màn hình của người dùng.

Lớp SelectCharacterView có nhiệm vụ hiển thị giao diện chọn nhân vật và bắt các sự kiện trên màn hình của người dùng.

Lớp MusicController có nhiệm vụ xử lý sự kiện tắt/bật nhạc.

- Biểu đồ tuần tự chức năng:



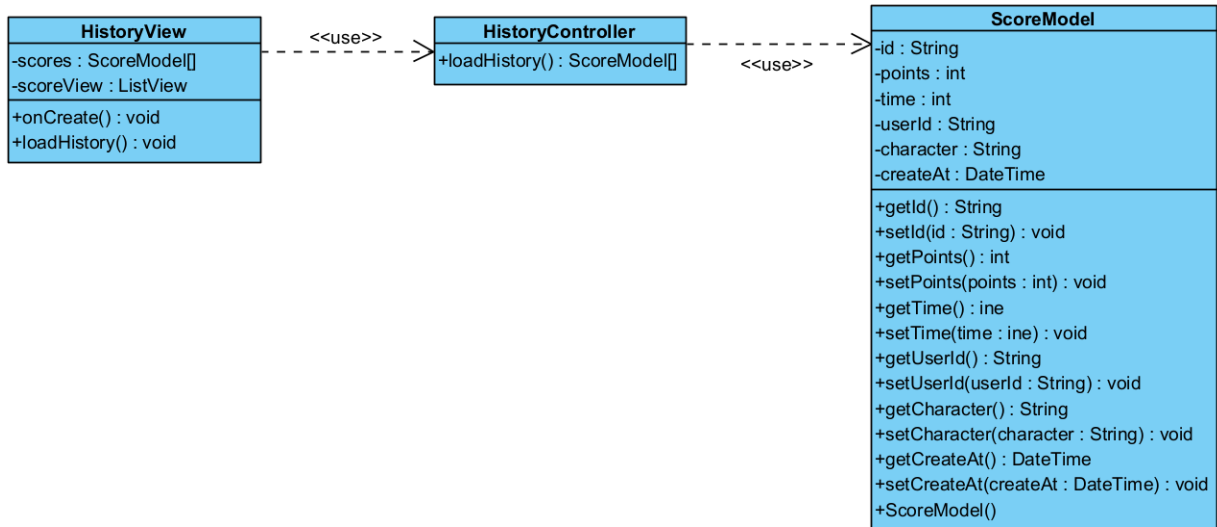
- 1) Người dùng mở ứng dụng game.
- 2) Lớp HomeView gọi hàm onCreate() khởi tạo giao diện màn trang chủ.
- 3) Hiển thị giao diện trang chủ lên màn hình.
- 4) Người dùng bấm vào biểu tượng loa.
- 5) HomeView bắt sự kiện và gọi hàm toggleMusic().
- 6) MusicController được gọi và gọi hàm toggleMusic() để tắt/bật âm nhạc trên phạm vi toàn cục ứng dụng.
- 7) Return về HomeView.

- 8) Thay đổi biểu tượng loa dựa trên trạng thái bật/tắt của nhạc.
 - 9) Người dùng bấm vào biểu tượng nhân vật để chuyển sang màn hình chọn nhân vật.
 - 10) HomeView bắt sự kiện và điều hướng sang màn chọn nhân vật.
 - 11) SelectCharacterView gọi hàm onCreate() để khởi tạo giao diện chọn nhân vật.
 - 12) Hiển thị giao diện chọn nhân vật lên màn hình.
 - 13) Người dùng bấm vào biểu tượng loa.
 - 14) SelectChracterView bắt sự kiện và gọi hàm toggleMusic().
 - 15) MusicController được gọi và gọi hàm toggleMusic() để thay đổi trạng thái bật/tắt của nhạc.
 - 16) Return về SelectCharacterView.
 - 17) SelectChracterView thay đổi biểu tượng loa dựa trên trạng thái bật/tắt của nhạc.
- Chức năng xem lịch sử chơi:
 - Giao diện chức năng:



04/11/2024	Mask Dude	130	50s
04/11/2024	Mask Dude	130	52s
04/11/2024	Mask Dude	0	9s
04/11/2024	Mask Dude	10	18s

- Biểu đồ lớp chi tiết:

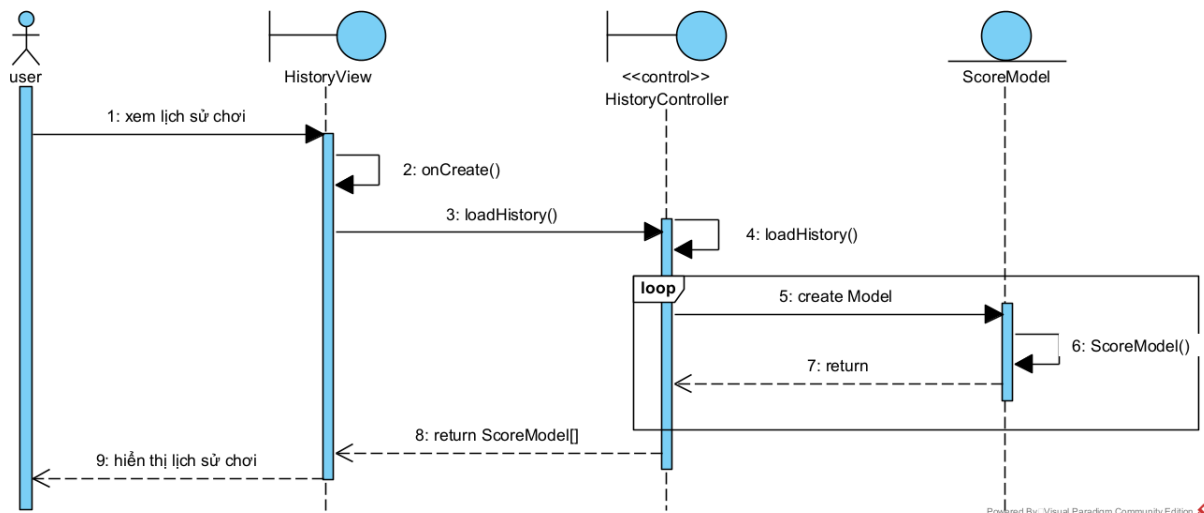


Lớp HistoryView có nhiệm vụ hiển thị giao diện lịch sử chơi lên màn hình và bắt các sự kiện trên màn hình của người dùng.

Lớp HistoryController có nhiệm vụ tải lịch sử chơi trên thiết bị của người dùng.

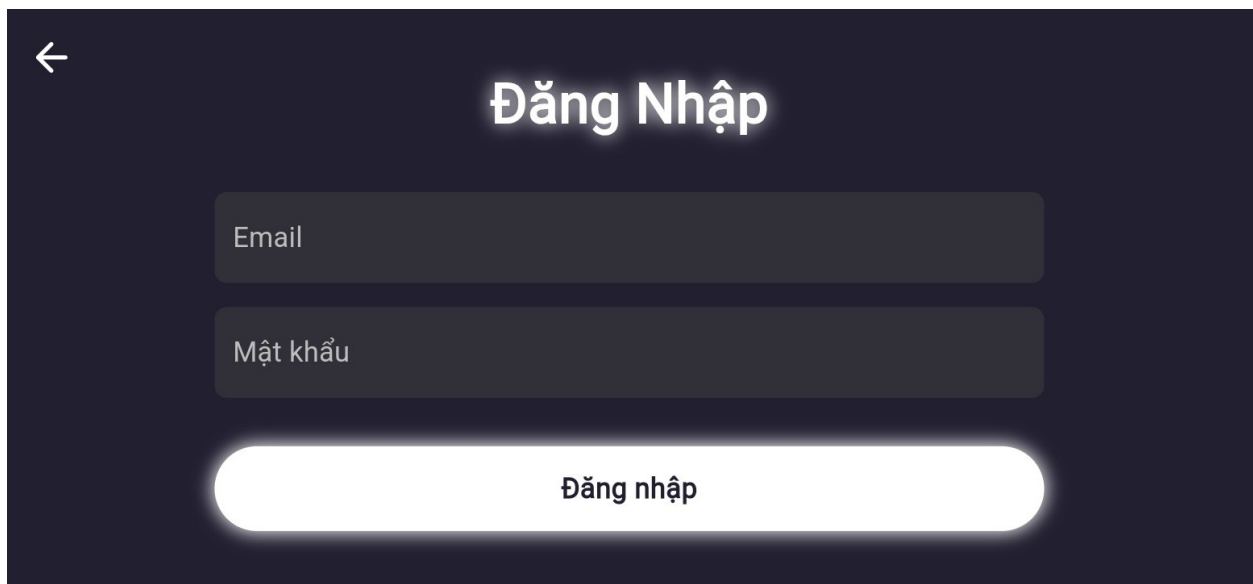
Lớp ScoreModel là lớp thực thể cho dữ liệu điểm của 1 lần chơi.

- Biểu đồ tuần tự chức năng:



- 1) Người dùng bấm xem lịch sử chơi.
- 2) HistoryView gọi hàm onCreate() khởi tạo giao diện.
- 3) HistoryView gọi hàm loadHistory().

- 4) HistoryController được gọi và gọi hàm loadHistory()
 - 5) Sử dụng dữ liệu đã tải để tạo ra các ScoreModel.
 - 6) Tạo đối tượng ScoreModel.
 - 7) Trả về đối tượng ScoreModel.
 - 8) Trả về danh sách đối tượng ScoreModel.
 - 9) Hiển thị lịch sử chơi dưới.
- Chức năng đăng nhập:
 - Giao diện chức năng:

A dark-themed login screen with a dark blue background. At the top left is a white back arrow. In the center, the text 'Đăng Nhập' is displayed in a large, bold, white font. Below this, there are two dark gray input fields with rounded corners. The first field is labeled 'Email' in a light gray font, and the second field is labeled 'Mật khẩu' in a light gray font. At the bottom, there is a large, white, rounded rectangular button with the text 'Đăng nhập' in a dark gray font.

←

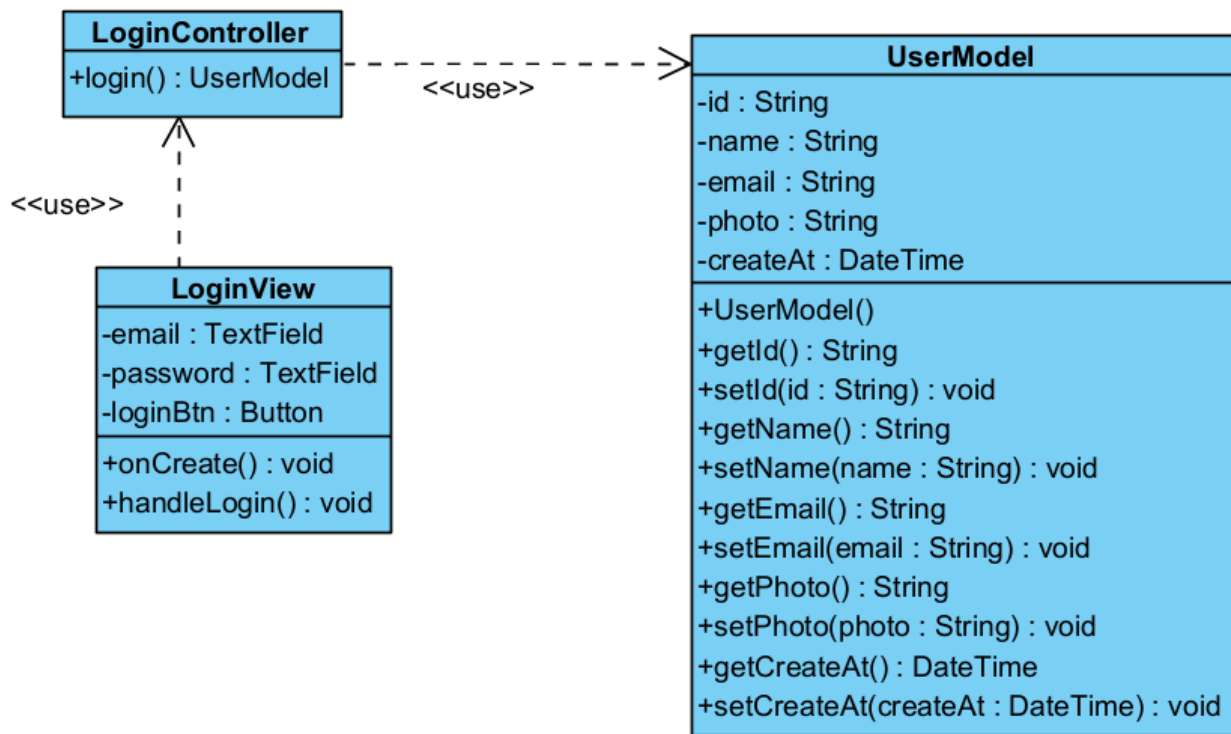
Đăng Nhập

Email

Mật khẩu

Đăng nhập

- Biểu đồ lớp chi tiết:

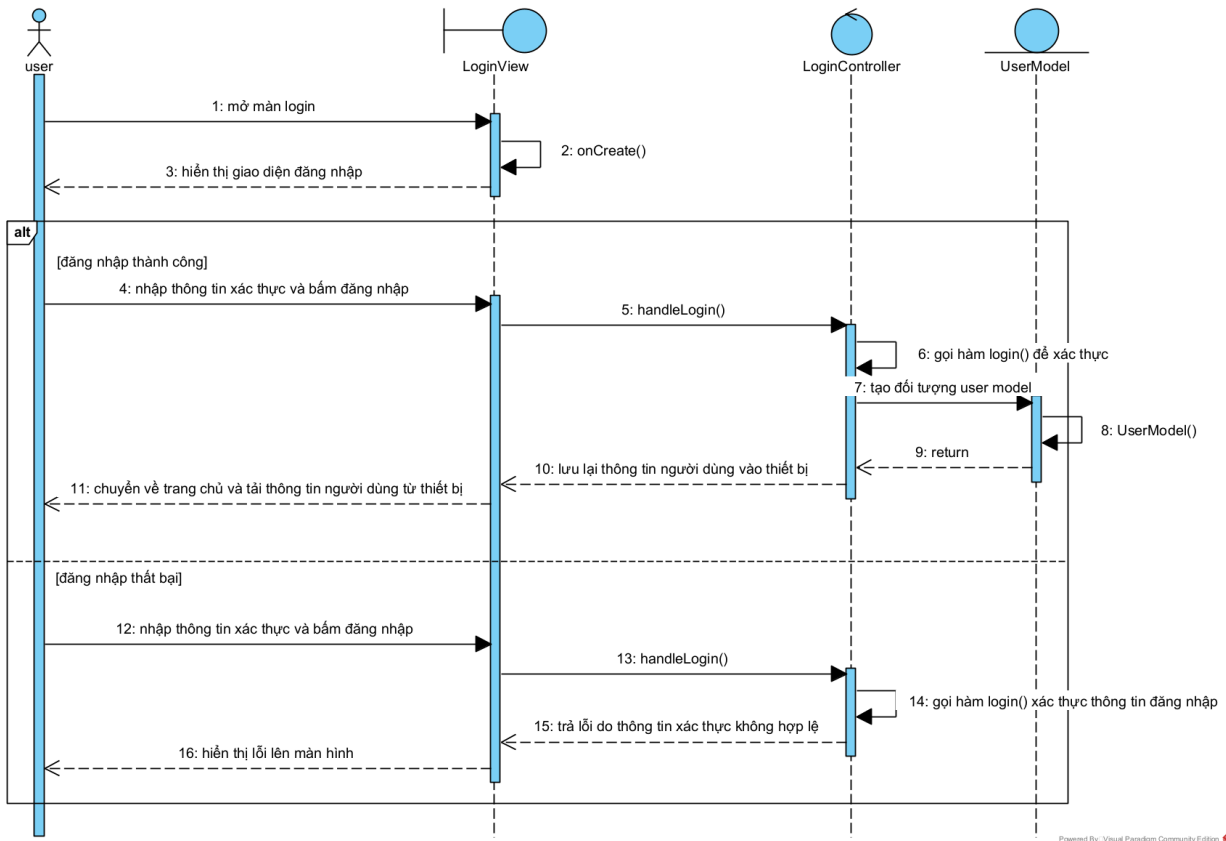


Lớp LoginView có nhiệm vụ hiển thị giao diện đăng nhập và lắng nghe các sự kiện trên màn hình của người dùng.

Lớp LoginController có nhiệm vụ xử lý sự kiện đăng nhập.

Lớp UserModel là lớp thực thể cho dữ liệu thông tin người dùng.


- Biểu đồ tuần tự chức năng:



- 1) Người dùng mở màn hình đăng nhập.
 - 2) LoginView gọi hàm onCreate() và khởi tạo giao diện màn hình đăng nhập.
 - 3) Hiển thị giao diện lên màn hình thiết bị của người dùng.
 - 4) Người dùng nhập thông tin xác thực và bấm đăng nhập.
 - 5) LoginView bắt sự kiện đăng nhập gọi hàm handleLogin().
 - 6) LoginController được gọi và gọi hàm login() để xác thực.
 - 7) Xác thực thành công và tạo đối tượng UserModel.
 - 8) Đối tượng UserModel được tạo.
 - 9) Trả về đối tượng UserModel.
 - 10) Lưu lại thông tin người dùng vào thiết bị.
 - 11) Chuyển về trang chủ và tải thông tin người dùng từ thiết bị.
- Chức năng đăng ký:
- Giao diện chức năng:

←

Đăng Ký



Tên hiển thị

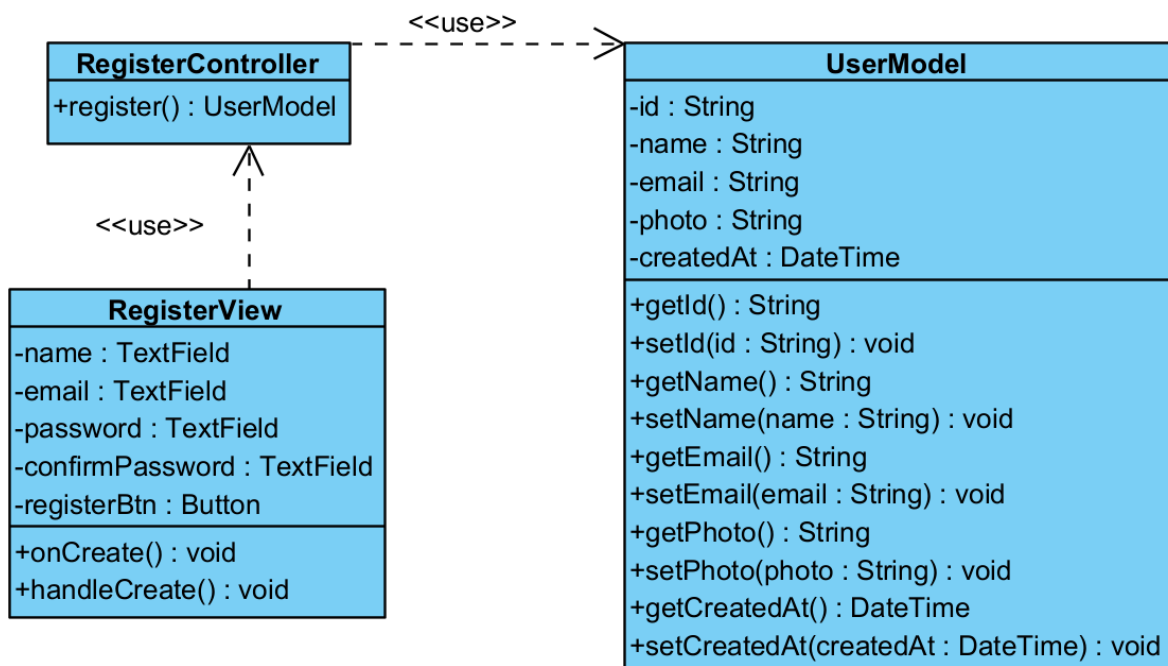
Email

Mật khẩu

Xác nhận mật khẩu

Đăng ký

- Biểu đồ lớp chi tiết:

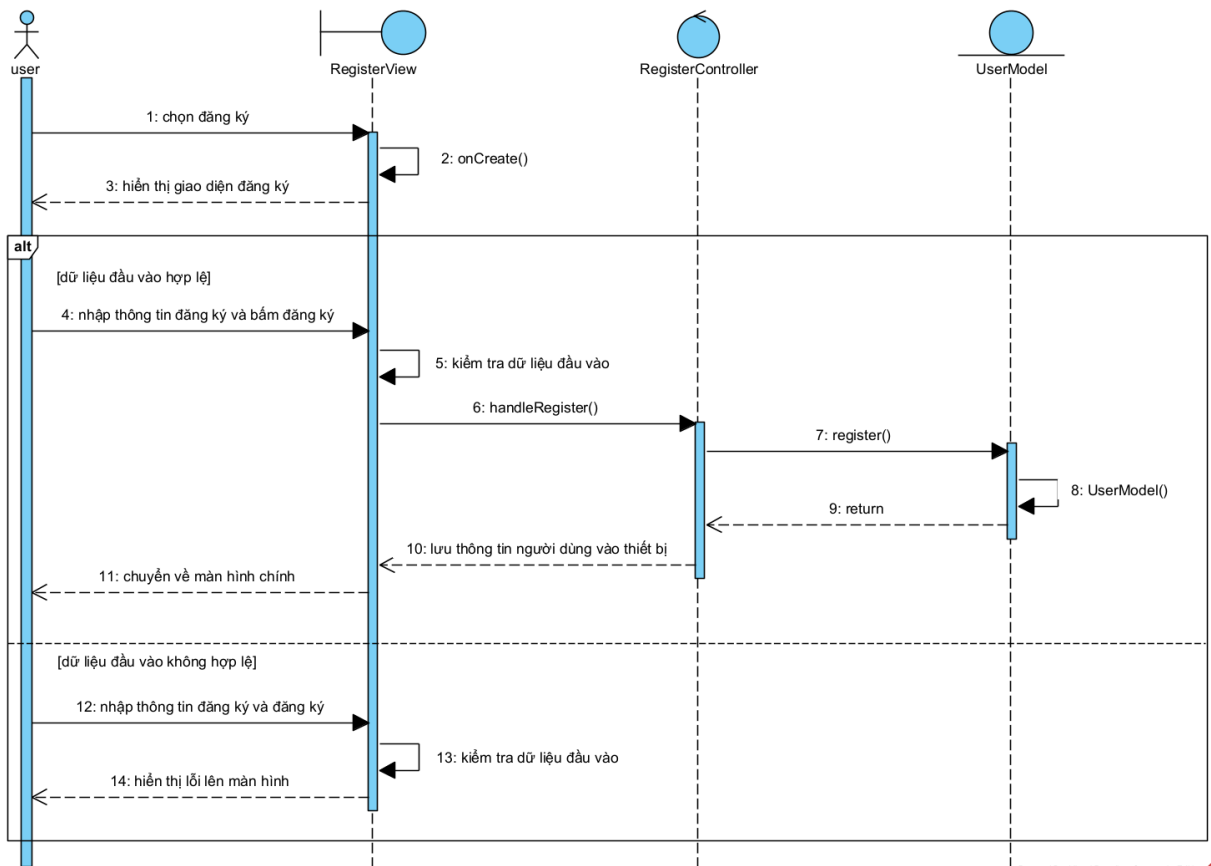


Lớp RegisterView có nhiệm vụ hiển thị giao diện đăng ký và lắng nghe các sự kiện trên màn hình.

Lớp RegisterController có nhiệm vụ xử lý sự kiện đăng ký của người dùng.

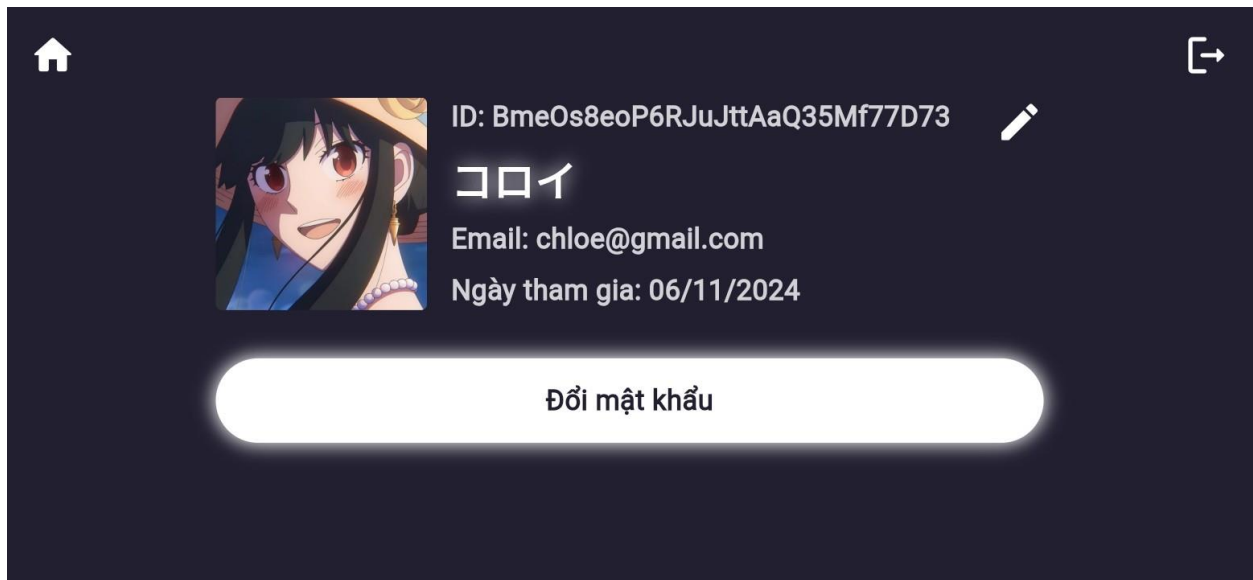
Lớp UserModel là lớp thực thể cho dữ liệu thông tin người dùng.

- Biểu đồ tuần tự chức năng:

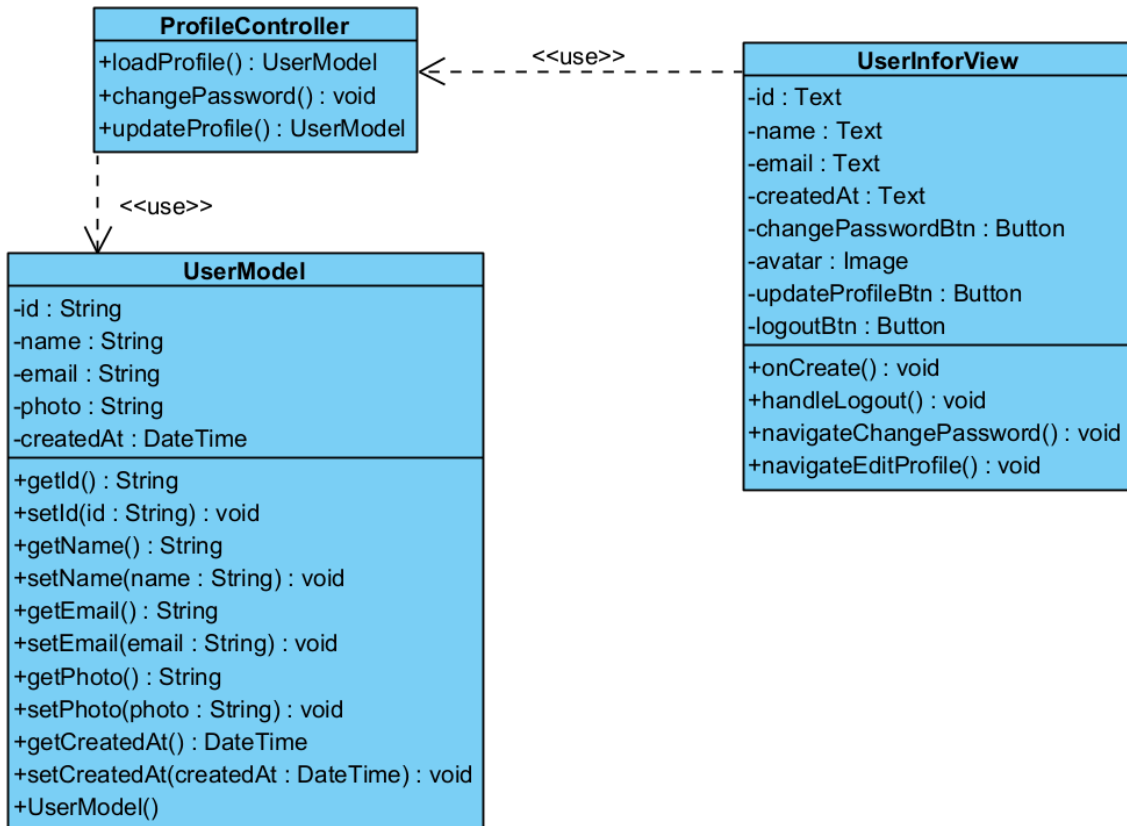


- 1) Người dùng chọn đăng ký.
- 2) RegisterView gọi hàm onCreate() khởi tạo giao diện đăng ký.
- 3) Hiển thị giao diện đăng ký lên màn hình của người dùng.
- 4) Người dùng nhập thông tin đăng ký và bấm đăng ký.
- 5) RegisterView kiểm tra dữ liệu đầu vào.
- 6) RegisterView gọi hàm handleRegister().
- 7) RegisterController được gọi và gọi hàm register() để xử lý sự kiện đăng ký.
- 8) Đăng ký thành công và tạo đối tượng UserModel.

- 9) Trả về đối tượng UserModel chứa thông tin tài khoản người dùng.
 - 10) Lưu thông tin người dùng vào thiết bị.
 - 11) Chuyển về màn hình chính.
 - 12) Người dùng nhập thông tin đăng ký và bấm đăng ký.
 - 13) RegisterView kiểm tra dữ liệu đầu vào.
 - 14) Hiển thị lỗi lên màn hình.
- Chức năng xem thông tin tài khoản:
 - Giao diện chức năng:



- Biểu đồ lớp chi tiết:

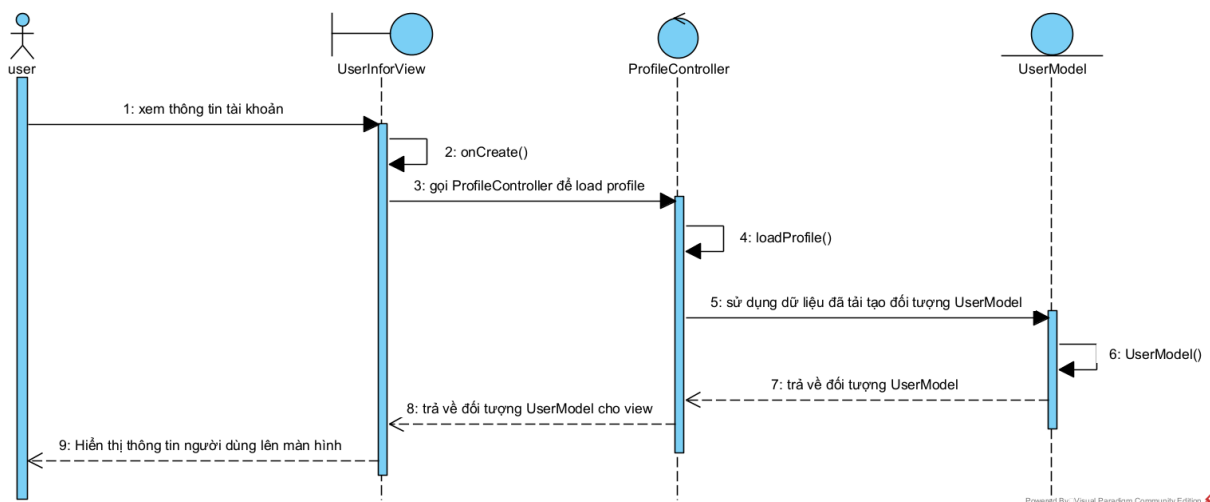


Lớp UserInforView có nhiệm vụ hiển thị giao diện thông tin người dùng và lắng nghe các sự kiện trên màn hình.

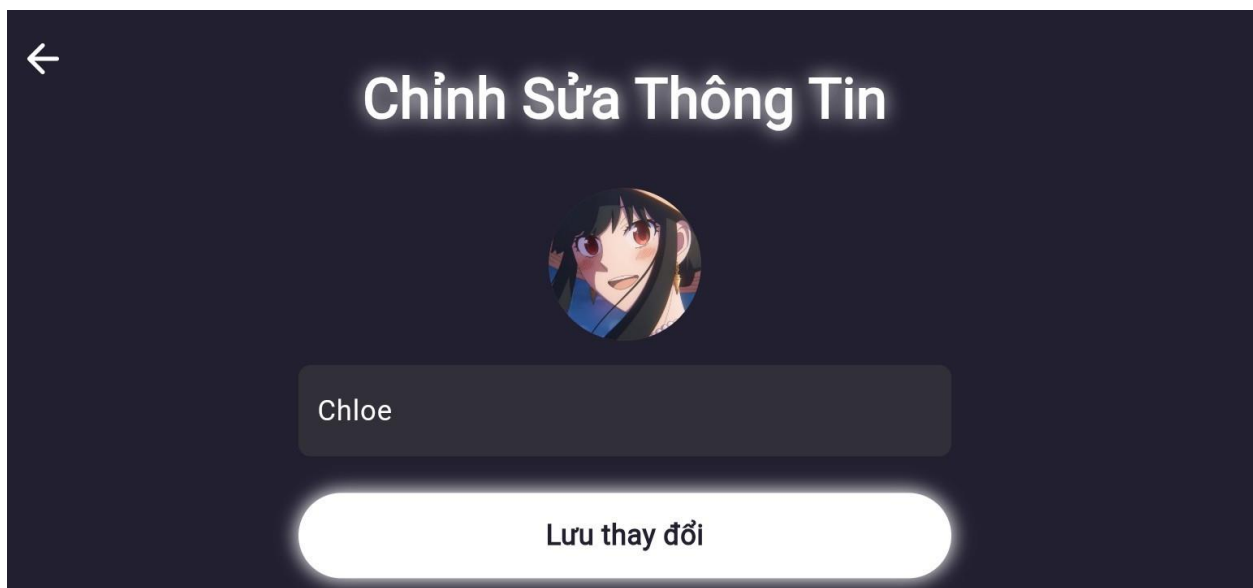
Lớp ProfileController có nhiệm vụ tải thông tin người dùng.

Lớp UserModel là lớp thực thể cho dữ liệu thông tin người dùng.

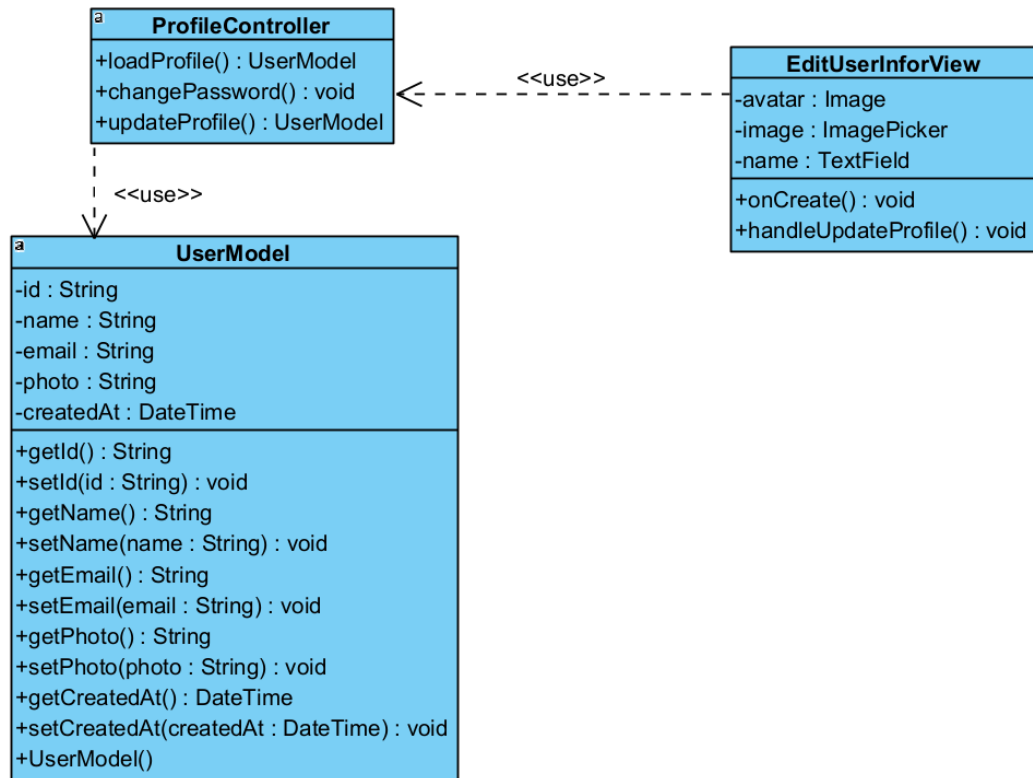
- Biểu đồ tuần tự chức năng:



- 1) Người dùng chọn xem thông tin tài khoản.
 - 2) UserInforView gọi hàm onCreate() khởi tạo giao diện.
 - 3) UserInforView gọi ProfileController để tải thông tin người dùng.
 - 4) ProfileController gọi hàm loadProfile().
 - 5) Sử dụng dữ liệu đã load để tạo đối tượng UserModel.
 - 6) Tạo đối tượng UserModel.
 - 7) Trả về đối tượng UserModel được tạo.
 - 8) Trả về đối tượng UserModel cho UserInforView.
 - 9) Hiển thị thông tin người dùng lên màn hình.
- Chức năng cập nhật thông tin tài khoản
 - Giao diện chức năng:



- Biểu đồ lớp chi tiết:

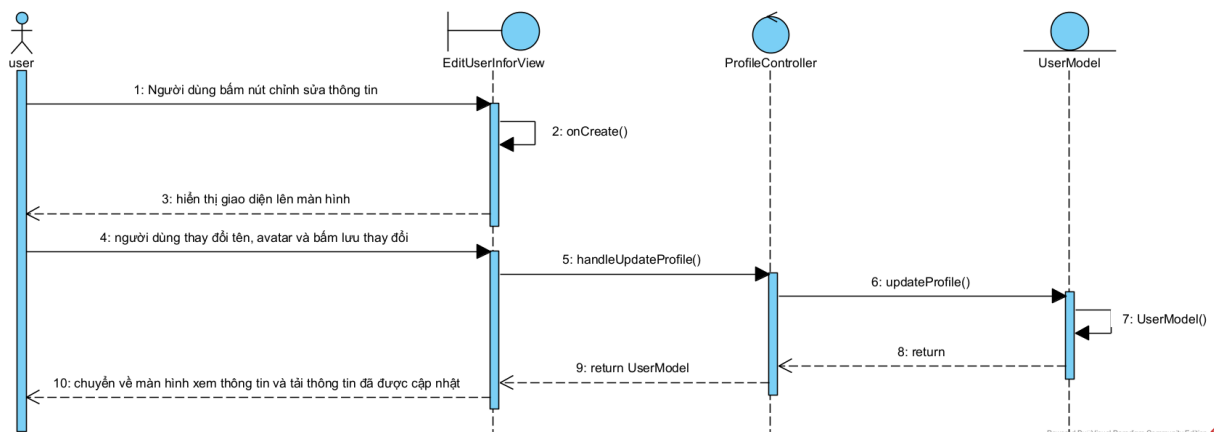


Lớp EditUserInforView có nhiệm vụ hiển thị giao diện cho phép sửa đổi tên và ảnh đại diện người dùng.

Lớp ProfileController có nhiệm vụ xử lý sự kiện cập nhật thông tin người dùng.

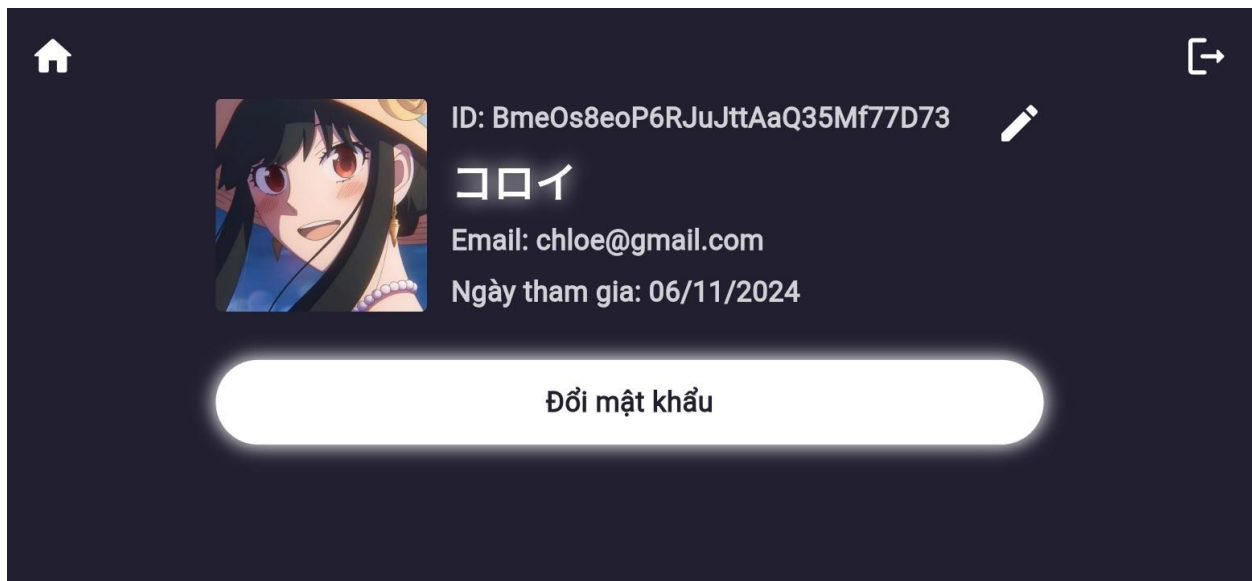
Lớp UserModel là lớp thực thể cho dữ liệu thông tin người dùng.

- Biểu đồ tuần tự chức năng:

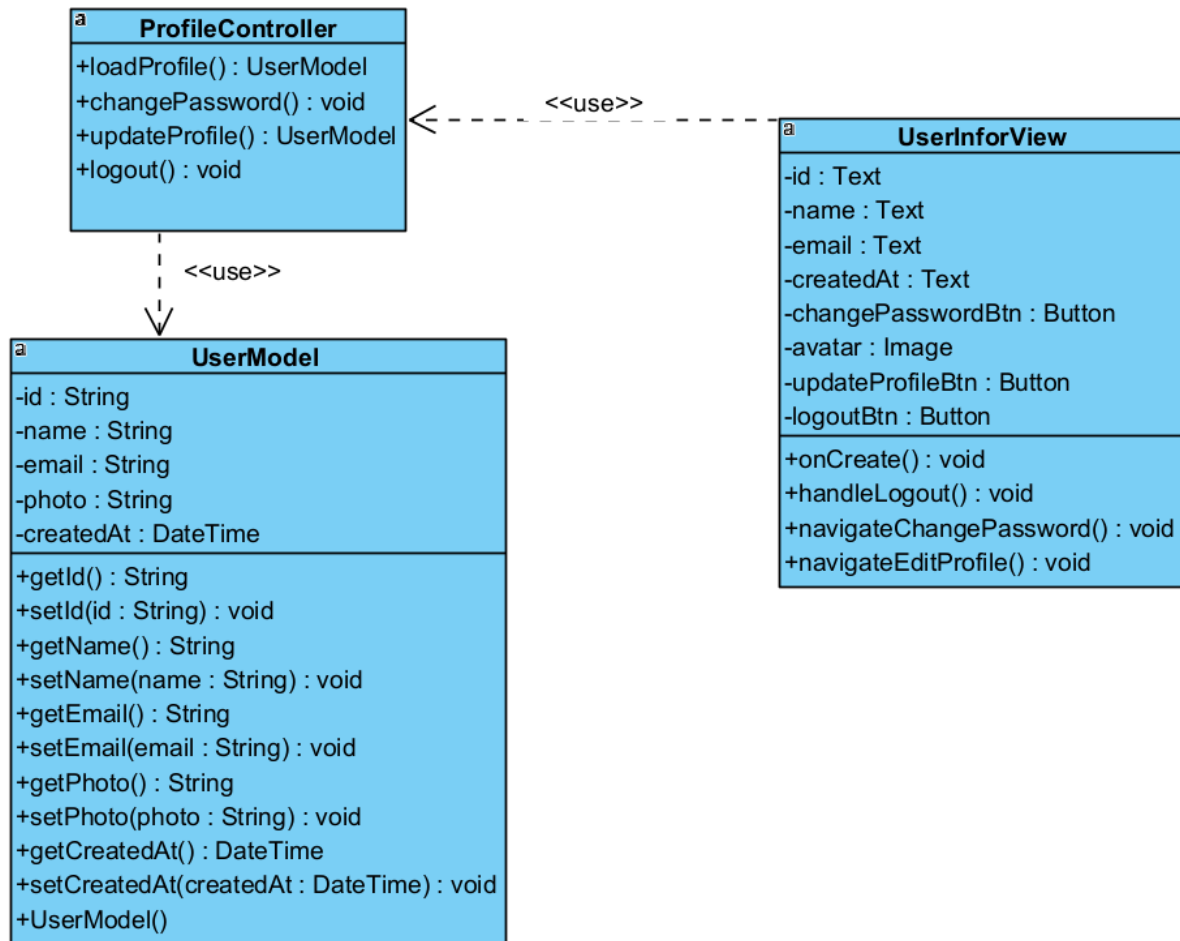


- 1) Người dùng bấm nút chỉnh sửa thông tin.

- 2) EditUserInfoView gọi hàm onCreate() để khởi tạo giao diện.
 - 3) Hiển thị giao diện lên màn hình với tên và avatar hiện tại.
 - 4) Người dùng thay đổi tên, avatar và bấm lưu thay đổi.
 - 5) EditUserInfoView bắt sự kiện và gọi hàm handleUpdateProfile().
 - 6) ProfileController được gọi và gọi hàm updateProfile().
 - 7) Tạo đối tượng UserModel từ dữ liệu mới.
 - 8) Trả về đối tượng UserModel.
 - 9) Trả đối tượng UserModel cho EditUserInfoView.
 - 10) Chuyển về màn hình xem thông tin người dùng và tải thông tin đã được cập nhật.
- Chức năng đăng xuất:
 - Giao diện chức năng:



- Biểu đồ lớp chi tiết:

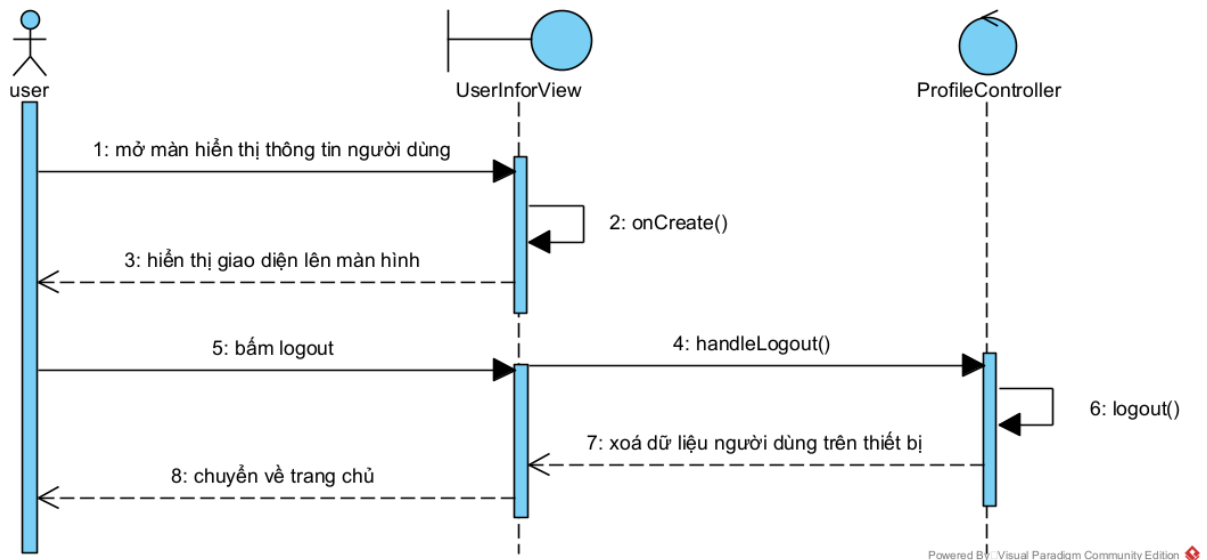


Lớp UserInforView có nhiệm vụ hiển thị giao diện thông tin người dùng và lắng nghe các sự kiện trên màn hình.

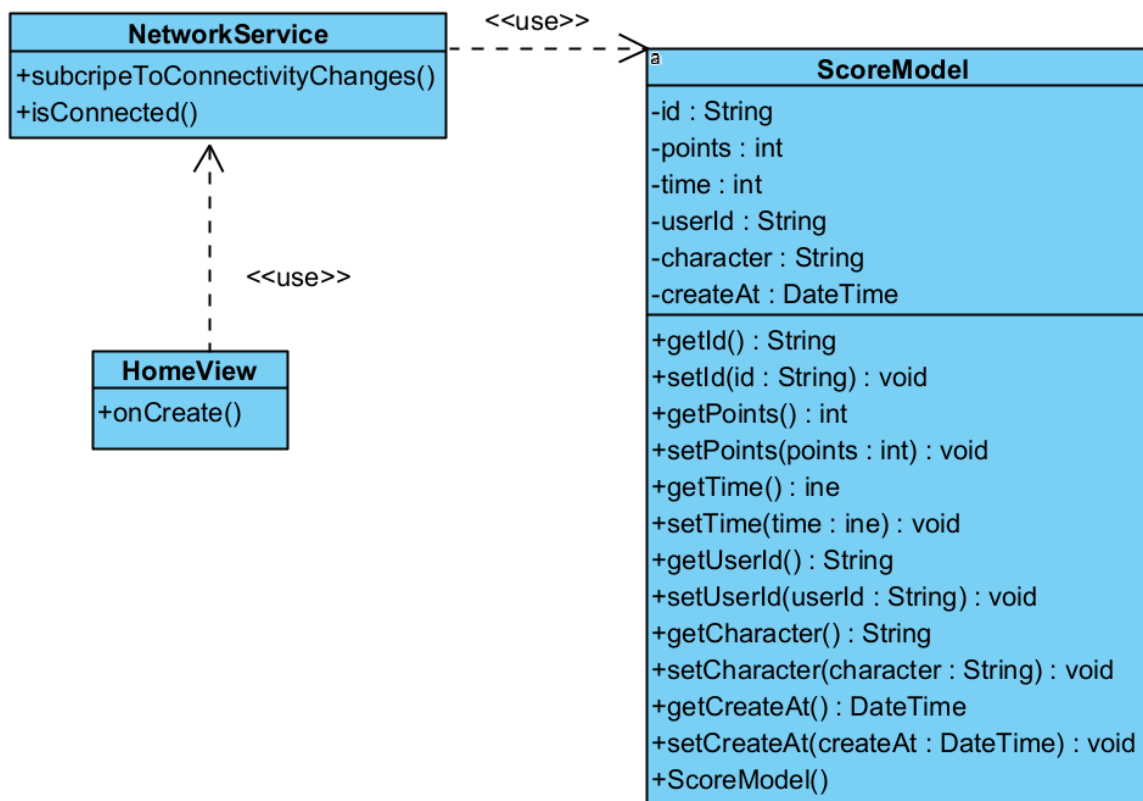
Lớp ProfileController có nhiệm vụ xử lý sự kiện đăng xuất.

Lớp UserModel là lớp thực thể cho dữ liệu thông tin người dùng.

- Biểu đồ tuần tự chức năng:



- 1) Người dùng mở màn hình hiển thị thông tin người dùng.
 - 2) UserInforView gọi hàm onCreate() để khởi tạo giao diện.
 - 3) Hiển thị giao diện lên màn hình.
 - 4) Người dùng bấm logout.
 - 5) UserInforView bắt sự kiện và gọi hàm handleLogout().
 - 6) ProfileController được gọi và gọi hàm logout().
 - 7) Xoá dữ liệu người dùng trên thiết bị.
 - 8) Chuyển về màn hình trang chủ.
- Chức năng đồng bộ lịch sử chơi:
- Giao diện chức năng: chức năng là một dịch vụ chạy ngầm, không có giao diện.
 - Biểu đồ lớp chi tiết:

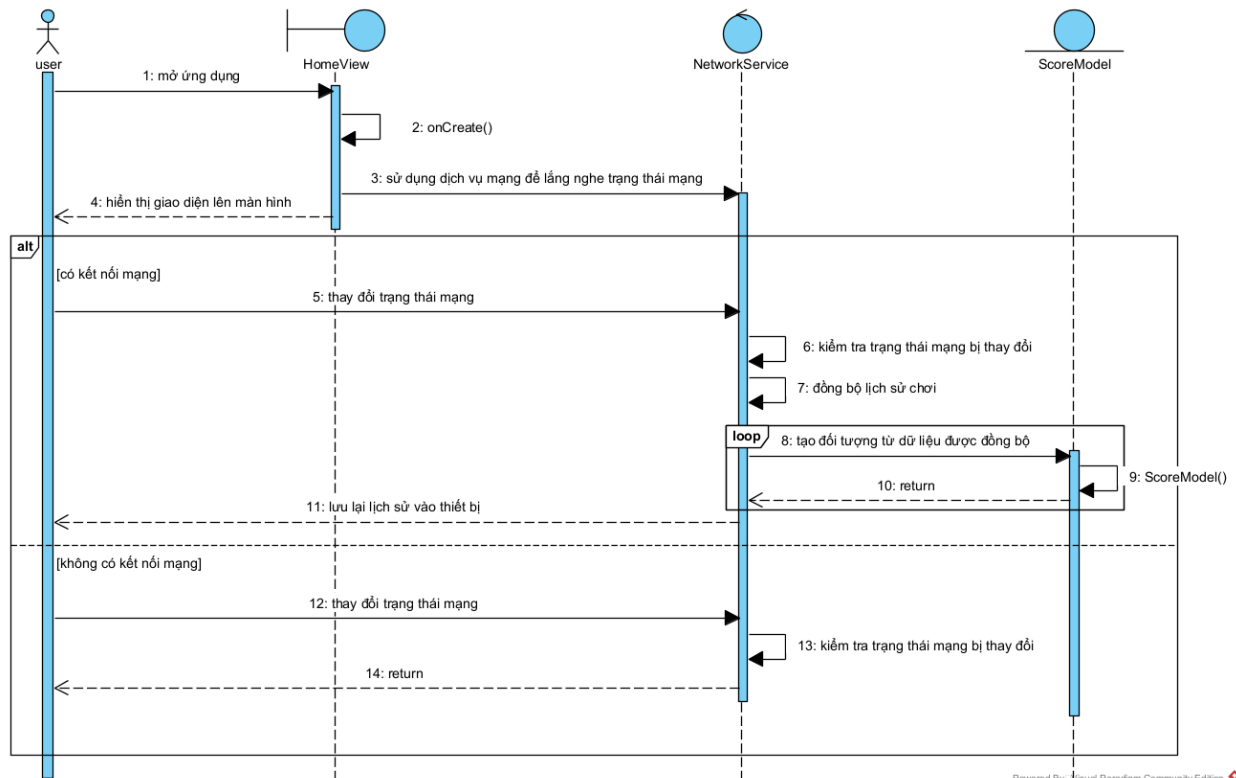


Lớp **HomeView** có nhiệm vụ hiển thị giao diện màn hình trang chủ và lắng nghe các sự kiện trên màn hình của người dùng.

Lớp **NetworkService** có nhiệm vụ lắng nghe sự thay đổi của trạng thái mạng để đồng bộ dữ liệu lịch sử từ thiết bị lên server và ngược lại.

Lớp **ScoreModel** là lớp thực thể cho dữ liệu điểm của 1 lần chơi.

- Biểu đồ tuần tự chức năng:




- 1) Người dùng mở ứng dụng.
 - 2) HomeView gọi hàm onCreate() khởi tạo giao diện.
 - 3) Gọi NetworkService để lắng nghe trạng thái mạng.
 - 4) Hiển thị giao diện lên màn hình.
 - 5) Người dùng thay đổi trạng thái mạng.
 - 6) NetworkService kiểm tra kết nối mạng bị thay đổi.
 - 7) Đồng bộ lịch sử chơi.
 - 8) Tạo đối tượng từ dữ liệu được đồng bộ.
 - 9) Đối tượng ScoreModel() được tạo.
 - 10) Trả về đối tượng ScoreModel().
 - 11) Lưu lại lịch sử vào thiết bị.
 - 12) Người dùng thay đổi trạng thái mạng.
 - 13) NetworkService kiểm tra kết nối mạng bị thay đổi.
 - 14) Không có gì xảy ra.
- Chức năng xem bảng xếp hạng:
 - Giao diện chức năng:



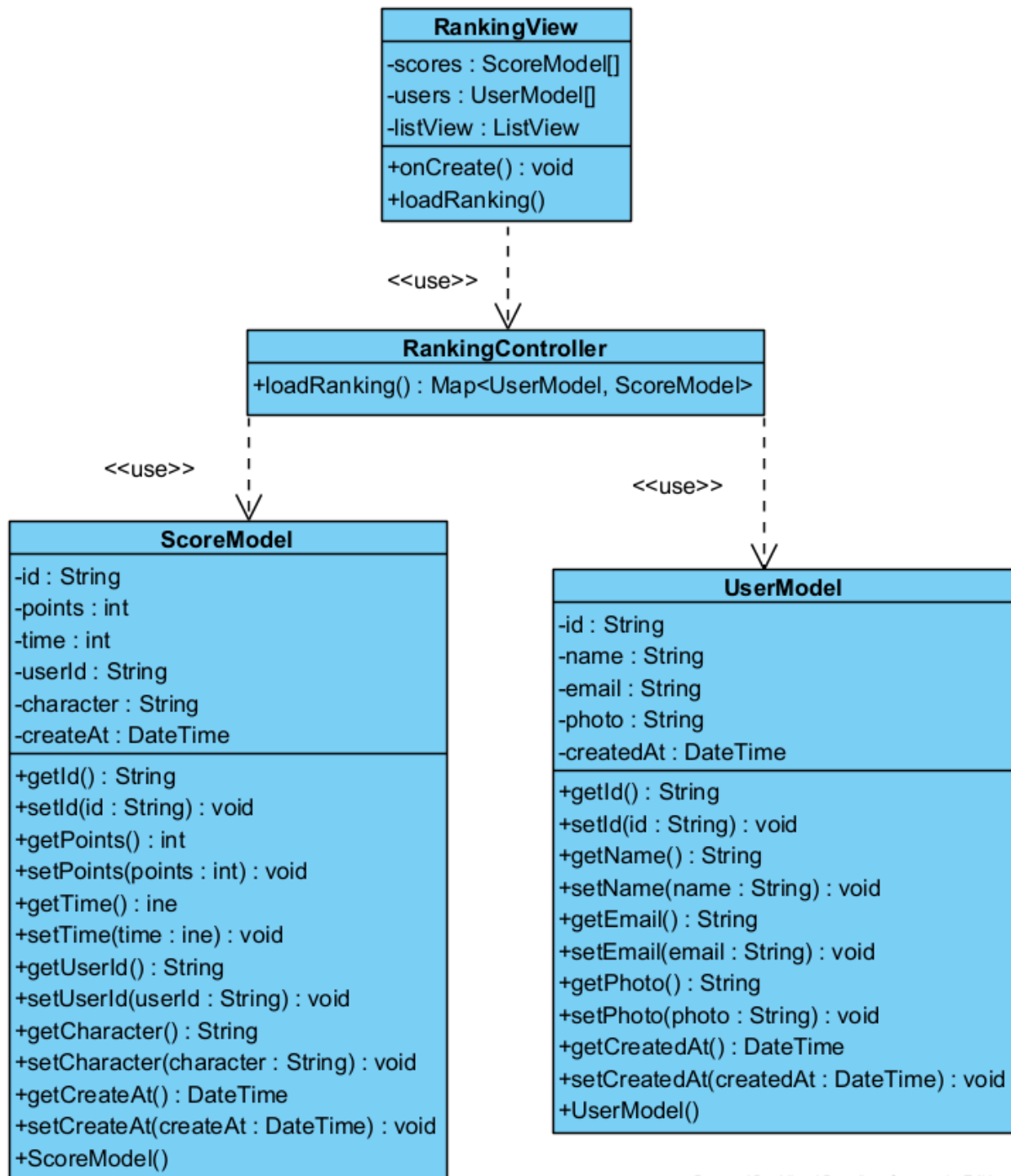
Bảng xếp hạng

	Tên	Điểm TB	Thời gian	Lượt chơi
1	Thẩm Ấu Sở	92.0	29.6s	5
2	a	16.7	9.3s	3
3	hshshs	25.0	12.0s	2
4	コロイ	10.0	10.5s	2



	Tên	Điểm TB	Thời gian	Lượt chơi
1	Thẩm Ấu Sở	92.0	29.6s	5
2	a	16.7	9.3s	3
3	hshshs	25.0	12.0s	2
4	コロイ	10.0	10.5s	2
5	testt	NaN	NaNs	0

- Biểu đồ lớp chi tiết:



Powered By: Visual Paradigm Community Edition

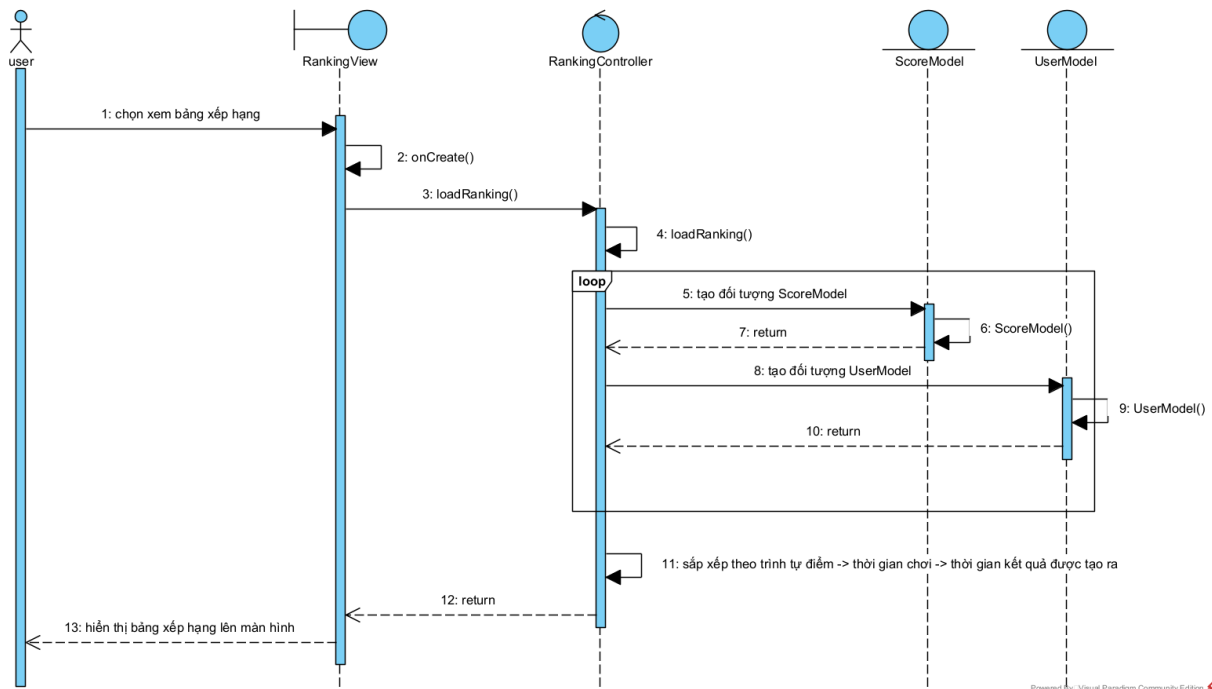
Lớp RankingView có nhiệm vụ hiển thị giao diện bảng xếp hạng lên màn hình và lắng nghe các sự kiện trên màn hình của người dùng.

Lớp RankingController có nhiệm vụ tải danh sách bảng xếp hạng.

Lớp UserModel là lớp thực thể cho thông tin dữ liệu của người dùng.

Lớp ScoreModel là lớp thực thể cho dữ liệu điểm của một lần chơi.

- Biểu đồ tuần tự chức năng:



- 1) Người dùng chọn xem bảng xếp hạng.
- 2) RankingView gọi onCreate() và khởi tạo giao diện.
- 3) RankingView gọi hàm loadRanking().
- 4) RankingController được gọi và gọi hàm loadRanking().
- 5) Tạo đối tượng ScoreModel từ dữ liệu.
- 6) Đối tượng ScoreModel được tạo.
- 7) Trả về đối tượng ScoreModel.
- 8) Tạo đối tượng UserModel.
- 9) Đối tượng UserModel được tạo.
- 10) Trả về đối tượng UserModel.
- 11) Sắp xếp theo trình tự điểm -> thời gian chơi -> thời gian kết quả được tạo ra.
- 12) Trả về danh sách bảng xếp hạng.
- 13) Hiển thị bảng xếp hạng lên màn hình.