**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

---□ & □---

A blue book with a ring around it

Description automatically generated

**BÁO CÁO**

**MÔN HỌC: LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG**

Đề tài:

**ỨNG DỤNG QUẢNG CÁO, BÁN SÁCH ONLINE**

**Sinh viên thực hiện: Hồ Ngọc Giang – 2051012024**

**Trần Duy Khang -2051012044**

**Lê Phi Long - 2051012056**

**Nguyễn Thị Ý Nhi - 2051012082**

**Trần Khải Quân – 2051012096**

**Phan Thị Hồng Quế - 2051012097**

**Giảng viên hướng dẫn: ThS. Lê Ngọc Hiếu**

**Thành phố Hồ Chí Minh, tháng 8 năm 2023**

**DANH SÁCH THÀNH VIÊN NHÓM VÀ PHÂN CÔNG CÔNG VIỆC**

|  |  |  |
| --- | --- | --- |
| **BẢNG PHÂN CÔNG LÀM VIỆC** | | |
| **Họ Và Tên** | **MSSV** | **Phân Công** |
| 1. Hồ Ngọc Giang | 2051012024 | Xây dựng cơ sở dữ liệu và thiết kế API |
| 2. Trần Duy Khang | 2051012044 | Xử lý chức năng thêm sản phẩm vào giỏ hàng, thiết kế giao diện danh mục lục. |
| 3. Lê Phi Long | 2051012056 | Thiết kế giao diện đăng ký, đăng nhập, trang chủ, danh mục, xử lý gọi API để hiển thị sản phẩm, tác giả, khuyến mãi và chức năng tìm kiếm |
| 4. Nguyễn Thị Ý Nhi | 2051012082 | Xử lý chức năng đăng ký, đăng nhập, trang hồ sơ cá nhân |
| 5. Trần Khải Quân | 2051012096 | Xử lý chức năng thêm sản phẩm vào giỏ hàng, thiết kế giao diện danh mục lục. |
| 6. Phan Thị Hồng Quế | 2051012097 | Xử lý chức năng đăng ký, đăng nhập, trang hồ sơ cá nhân |
| **\*Note:** Bên cạnh những công việc chính được phân công thì các thành viên trong nhóm vẫn tham gia đóng góp ý kiến đầy đủ. | | |

MỤC LỤC

[1. Mô tả nghiệp vụ 4](#_Toc144219855)

[1.1. Đăng ký 4](#_Toc144219856)

[1.2. Đăng nhập 4](#_Toc144219857)

[1.3. Tìm kiếm sách 4](#_Toc144219858)

[1.4. Hiển thị slide quảng cáo 4](#_Toc144219859)

[1.5. Hiển thị trang tác giả nổi bật và các sách nổi bật 4](#_Toc144219860)

[1.6. Hiển thị các khuyến mãi, khuyến mãi theo phần trăm 5](#_Toc144219861)

[1.7. Lọc sách theo thể loại ở danh mục 5](#_Toc144219862)

[1.8. Tùy chỉnh thông tin hồ sơ cá nhân 5](#_Toc144219863)

[1.9. Thêm sách vào giỏ hàng 5](#_Toc144219864)

[2. Tài liệu kỹ thuật: 6](#_Toc144219865)

[2.1. Thiết kế cơ sở dữ liệu và API 6](#_Toc144219866)

[2.2. Chức năng đăng ký, đăng nhập 11](#_Toc144219867)

[2.3. Chức năng hiển thị slide quảng cáo sản phẩm 18](#_Toc144219868)

[2.4. Chức năng hiển thị tác giả, sách, khuyến mãi 21](#_Toc144219869)

[2.5. Chức năng tìm kiếm sản phẩm 25](#_Toc144219870)

[2.6. Chức năng sắp xếp sách theo thể loại trong danh mục 28](#_Toc144219871)

[2.7. Chức năng tùy chỉnh thông tin hồ sơ cá nhân 31](#_Toc144219872)

[2.8. Chức năng thêm sách vào giỏ hàng 33](#_Toc144219873)

[3. Hướng dẫn sử dụng: 35](#_Toc144219874)

[3.1. Đăng ký, đăng nhập 35](#_Toc144219875)

[3.2. Chỉnh sửa hồ sơ 35](#_Toc144219876)

[3.3. Chọn Sản phẩm và Thêm vào Giỏ hàng 36](#_Toc144219877)

[3.4. Quản lý giỏ hàng 36](#_Toc144219878)

[3.5. Thanh toán 36](#_Toc144219879)

## 1. Mô tả nghiệp vụ

### 1.1. Đăng ký

Register là quá trình cho phép người dùng tạo tài khoản cá nhân trên ứng dụng để có thể sử dụng các tính năng và dịch vụ của ứng dụng. Nghiệp vụ này bao gồm:

**Nhập thông tin cá nhân:** Trên màn hình Đăng ký, người dùng sẽ được yêu cầu cung cấp thông tin cần thiết để tạo tài khoản. Thông tin bao gồm: Tên tài khoản, địa chỉ email, mật khẩu, nhập lại mật khẩu.

**Xác nhập đăng ký:** Người dùng chọn nút “Đăng ký” sau khi đã kiểm tra thông tin và chắc chắn rằng mọi thông tin đều chính xác.

**Tạo tài khoản thành công:** Sau khi người dùng đã xác nhận và tạo tài khoản thành công, ứng dụng sẽ cung cấp thông báo và chuyển hướng đến màn hình Đăng nhập để người dùng có thể đăng nhập bằng tài khoản vừa tạo.

**Thông báo lỗi (nếu có):** Trong trường hợp người dùng không nhập đầy đủ thông tin hoặc thông tin không hợp lệ, ứng dụng sẽ hiện thị thông báo lỗi và yêu cầu người dùng điều chỉnh.

### 1.2. Đăng nhập

Login là quá trình cho phép người dùng truy cập và tài khoản cá nhân của mình bằng cách nhập thông tin đăng nhập đã đăng ký trước đó. Nghiệp vụ này bao gồm:

**Nhập thông tin đăng nhập:** Người dùng sẽ được yêu cầu nhập thông tin đăng nhập để truy cập và tài khoản của mình. Thông tin này bao gồm: địa chỉ email và mật khẩu đã đăng ký tài khoản.

**Xác nhận đăng nhập:** Người dùng chọn nút “Đăng nhập” sau khi nhập đúng thông tin đăng nhập.

**Xác thực thông tin:** Ứng dụng sẽ kiểm tra thông tin đăng nhập và mật khẩu để xác nhận tính chính xác. Nếu thông tin không khớp, người dùng sẽ nhận thông báo lỗi.

**Đăng nhập thành công:** Trong trường hợp đúng thông tin thì sẽ chuyển hướng đến màn hình chính của app bán sách.

### 1.3. Tìm kiếm sách

Người dùng có thể dùng để search tên sách mà mình mong muốn để tìm nhanh chóng và tiết kiệm thời gian.

### 1.4. Hiển thị slide quảng cáo

Sẽ có slide quảng cáo hiện ở trang chủ để quảng cáo các loại sự kiện, sách nổi bật.

### 1.5. Hiển thị trang tác giả nổi bật và các sách nổi bật

Hiển thị tên các giả nổi bật và các sách nổi bật ở trang chủ để khách hàng có thể thấy khi mới vào sử dụng ứng dụng.

### 1.6. Hiển thị các khuyến mãi, khuyến mãi theo phần trăm

Trong mỗi sách sẽ có giá khuyến mãi đi kèm, ở đây là giá khuyến mãi nằm kế giá gốc của sách.

### 1.7. Lọc sách theo thể loại ở danh mục

Danh mục là các phân loại chính của sách dựa trên các tiêu chí như thể loại, tác giả, giá cả, hình ảnh minh hoạ v.v. Nhiệm vụ của phần này bao gồm:

**Hiển thị danh sách sách:** Danh mục sẽ hiển thị danh sách các loại sách hoặc các chủ đề cụ thể để khách hàng có thể tìm kiếm và duyệt qua.

**Thanh tìm kiếm:** Người dùng có thể dùng để search tên sách mà mình mong muốn để tìm nhanh chóng và tiết kiệm thời gian.

**Chọn danh mục cụ thể:** Khi người dùng chọn một danh mục cụ thể, họ sẽ được dẫn đến trang danh sách các cuốn sách thuộc danh mục đó.

### 1.8. Tùy chỉnh thông tin hồ sơ cá nhân

Edit Proflie là quá trình cho phép người dùng cập nhật và điều chỉnh các thông tin cá nhân của họ trên tài khoản. Nhiệm vụ của phần này bao gồm:

**Hiển thị thông tin:** Trước khi chỉnh sửa, thông tin cá nhân hiện tại của người dùng sẽ hiện thị những thông tin đã có khi đăng ký, còn những chỗ chưa có thông tin sẽ để trống.

**Chỉnh sửa thông tin:** Người dùng có thể điều chỉnh các thông tin cá nhân sau: Tên đăng nhập, thay đổi mật khẩu, thay đổi số điện thoại, thay đổi email, giới tính, ngày sinh.

**Xác nhận lưu:** Người dùng khi đã điều chỉnh thông tin cá nhân chính xác nhấn nút “Save” thì hệ thống sẽ lưu lại các thông tin sau khi được chỉnh sửa.

### 1.9. Thêm sách vào giỏ hàng

Giỏ hàng là nơi mà người dùng có thể xem sách họ đã chọn mua và tổng số tiền họ phải trả. Nghiệp vụ của phần này bao gồm:

**Thêm và xóa sản phẩm**: Người dùng có thể thêm sách vào giỏ hàng bằng cách nhấn vào nút "Mua" từ trang sản phẩm. Họ cũng có thể xóa các sản phẩm ra khỏi giỏ hàng nếu cần.

**Tính toán tổng tiền:** Hệ thống sẽ tính toán tổng số tiền dựa trên số lượng và giá của từng cuốn sách trong giỏ hàng.

**Kiểm tra thông tin:** Người dùng có thể kiểm tra lại các sản phẩm trong giỏ hàng và tổng số tiền trước khi tiến hành thanh toán.

**Thanh toán**: Sau khi kiểm tra thông tin, người dùng có thể tiến hành thanh toán

Tổng cộng, cả "category" và "cart" đều có mục tiêu tạo ra trải nghiệm mua sắm thuận tiện, trực quan và thú vị cho khách hàng, từ việc duyệt sách đến quản lý đơn hàng.

## 2. Tài liệu kỹ thuật:

### 2.1. Thiết kế cơ sở dữ liệu và API

#### **2.1.1. Tổng quan về Spring Boot**

Trong báo cáo này, chúng ta sẽ tìm hiểu về cách xây dựng một API sử dụng Spring Boot và MySQL. Spring Boot là một framework phát triển ứng dụng Java được sử dụng rộng rãi để xây dựng các dịch vụ web. MySQL là một hệ quản trị cơ sở dữ liệu phổ biến được sử dụng để lưu trữ dữ liệu.

##### **2.1.1.1. Mô hình Spring MVC**

- Bao gồm 3 thành phần chính:

* Controller: Giữ nhiệm vụ điều hướng các yêu cầu từ người dùng, chứa logic doanh nghiệp, đảm nhận yêu cầu từ phía client, lưu trữ dữ liệu vào database, web service,…
* Model: Chứa các nhiệm vụ logic, phương thức xử lý, truy xuất database, hàm xử lý, class,…
* View: hiển thị thông tin tương tác với người dùng, là nơi tập hợp các form hoặc file HTML, CSS, template,…

A diagram of a database

Description automatically generated

*Hình 1.1 Mô hình MVC*

##### **2.1.1.2 Giới thiệu Spring Boot**

- Spring boot là Project nằm trong tầng IO Execution của Spring MVC Framework.

- Spring boot được tạo ra để dễ dàng hơn trong việc lập trình và phát triển ứng dụng. Spring boot hỗ trợ các bộ chứa nhúng, cho phép các ứng dụng web có thể chạy độc lập mà không cần triển khai ở các Web Server

- Như vậy Spring Boot bao gồm Spring Framework (Spring MVC) và Embedded HTTP server (Tomcat, servlet,…). Lợi ích của Spring boot là ta không cần phải cấu hình XML, giảm thiểu tối đa thời gian và năng suất.

##### **2.1.1.3 Khởi tạo dự án Spring Boot**:

- Để bắt đầu, ta cần khởi tạo một dự án Spring Boot. Bạn có thể sử dụng Spring Initializr hoặc các công cụ tương tự để tạo một dự án Spring Boot cơ bản.

##### **2.1.1.4 Cấu hình kết nối đến cơ sở dữ liệu MySQL:**

Trong tệp application.properties hoặc application.yml, ta cần cung cấp thông tin kết nối đến cơ sở dữ liệu MySQL, bao gồm URL, username và password.

A screen shot of a computer

Description automatically generatedVí dụ cấu hình trong tệp application.properties:

##### **2.1.1.5 Tạo Entity và Repository:**

Sử dụng các annotation như @Entity, @Table, @Id, @Column để định nghĩa các đối tượng Entity trong ứng dụng của bạn. Đây là các lớp Java đại diện cho các bảng trong cơ sở dữ liệu.

A screen shot of a computer

Description automatically generatedSử dụng @Repository để đánh dấu các interface Repository, chứa các phương thức để tương tác với cơ sở dữ liệu.

Ví dụ:

Hình 1.5.1: Tạo @Entity

A screen shot of a computer code

Description automatically generatedHình 1.5.2: Tạo @Repository

##### **2.1.1.6 Tạo các dịch vụ (Services):**

Tạo các lớp Service để xử lý logic kinh doanh và gọi các phương thức từ Repository để tương tác với cơ sở dữ liệu.

A screenshot of a computer program

Description automatically generatedVí dụ:

Hình 1.6: Lớp Service của Book

##### **2.1.1.7 Sử dụng các phương thức CRUD:**

- Spring Boot cung cấp các phương thức CRUD sẵn có thông qua JpaRepository. Bạn có thể sử dụng các phương thức như save(), findById(), findAll(), deleteById(), vv. để tương tác với cơ sở dữ liệu.

-Ví dụ:

A computer screen shot of text

Description automatically generatedHình 1.7: Controller sử dụng các phương thức CRUD

##### **2.1.1.8. Các phương thức API được sử dụng cho bài tập lớn.**

* A computer screen shot of text

  Description automatically generatedLấy danh sách tất cả sách: <https://book-api.up.railway.app/api/book/getAll>
* A screen shot of a computer code

  Description automatically generatedTìm kiếm sách theo Id: <https://book-api.up.railway.app/api/book/2>
* A screen shot of a computer program

  Description automatically generatedLấy tất cả danh mục: https://book-api.up.railway.app/api/category/getAll
* A computer screen with text

  Description automatically generatedTìm kiếm theo danh mục: https://book-api.up.railway.app/api/book/getByCategory/3222

#### **2.1.2. Tổng quan về MySQL Workbench**

##### **2.1.2.1. Giới thiệu:**

MySQL Workbench là một công cụ quản lý cơ sở dữ liệu mã nguồn mở, được phát triển bởi Oracle. Nó cung cấp một giao diện đồ họa để thiết kế cơ sở dữ liệu, quản lý và tương tác với cơ sở dữ liệu MySQL. Báo cáo này sẽ giới thiệu về MySQL Workbench và các tính năng chính của nó.

##### **2.1.2.2. Các tính năng chính của MySQL Workbench:**

- MySQL Workbench cung cấp một công cụ thiết kế ER (Entity-Relationship) cho phép bạn tạo, chỉnh sửa và quản lý cấu trúc cơ sở dữ liệu. Bạn có thể tạo bảng, quan hệ, chỉ mục và ràng buộc khóa ngoại trong môi trường đồ họa trực quan.

- MySQL Workbench cho phép bạn thực thi truy vấn SQL và tương tác với cơ sở dữ liệu MySQL. Bạn có thể viết và chạy các truy vấn, tạo, chỉnh sửa và xóa dữ liệu, và xem kết quả truy vấn trong một giao diện đồ họa thân thiện.

- Workbench cung cấp các công cụ để quản lý người dùng, vai trò và phân quyền trong cơ sở dữ liệu MySQL. Bạn có thể tạo người dùng mới, cấp và rút vai trò, thiết lập quyền truy cập cho từng người dùng và kiểm soát quyền truy cập vào dữ liệu.

- MySQL Workbench cho phép bạn thực hiện sao lưu và phục hồi cơ sở dữ liệu. Bạn có thể tạo bản sao lưu của toàn bộ cơ sở dữ liệu hoặc chỉ các bảng cụ thể, và khôi phục dữ liệu từ các bản sao lưu đã tạo.

- Workbench cung cấp các công cụ để xem và quản lý trạng thái hệ thống MySQL, bao gồm thông tin về phiên bản MySQL, tài nguyên hệ thống, quá trình và kết nối hiện tại.

##### **A screenshot of a computer Description automatically generated2.1.2.3. Mô hình ERD được sử dụng trong hệ thống**.

Hình 2.3: Mô hình ERD

#### **2.1.3. Triển khai ứng dụng bằng Railway**

##### **2.1.3.1. Giới thiệu**

Railway là một nền tảng triển khai ứng dụng dựa trên đám mây, giúp các nhà phát triển xây dựng, triển khai và quản lý ứng dụng một cách dễ dàng. Báo cáo này sẽ trình bày quy trình triển khai ứng dụng bằng Railway và các lợi ích của việc sử dụng nền tảng này.

##### **2.1.3.2. Quy trình triển khai bằng railway**

* Đăng ký và tạo dự án:

- Đăng ký tài khoản trên Railway (https://railway.app/) nếu chưa có.

- Tạo một dự án mới trên Railway để bắt đầu quá trình triển khai.

* Kết nối dự án với kho lưu trữ (Repository):

- Kết nối dự án với kho lưu trữ của bạn trên GitHub hoặc GitLab.

- Cho phép Railway truy cập vào kho lưu trữ để tự động triển khai từ mã nguồn.

* Thiết lập môi trường:

- Tạo một môi trường mới trong dự án và đặt tên cho nó.

- Cấu hình các biến môi trường cần thiết, chẳng hạn như cấu hình cơ sở dữ liệu, biến bí mật, v.v.

* Cấu hình tệp railway.yaml:

- Tạo hoặc chỉnh sửa tệp railway.yaml trong kho lưu trữ của bạn.

- Xác định các bước cần thiết để triển khai ứng dụng, bao gồm cài đặt phụ thuộc, xây dựng, chạy lệnh, v.v. Sử dụng cú pháp YAML để mô tả các bước này.

* Triển khai ứng dụng:

- Điều hướng đến trang dự án trên Railway.

- Chọn môi trường bạn đã tạo và nhấp vào nút "Deploy" để bắt đầu quá trình triển khai.

- Railway sẽ tự động lấy mã nguồn từ kho lưu trữ và triển khai ứng dụng dựa trên cấu hình trong railway.yaml.

* Theo dõi tiến trình triển khai:

- Railway cung cấp giao diện để bạn theo dõi tiến trình triển khai và xem các bước đang được thực thi.

- Bạn có thể xem các lỗi, thông báo và nhật ký từ quá trình triển khai để xác định và sửa chữa các vấn đề (nếu có).

* Kiểm tra ứng dụng:

- Khi quá trình triển khai hoàn thành, bạn có thể truy cập vào ứng dụng của mình bằng đường dẫn được cung cấp bởi Railway.

##### **2.1.3.3. Lợi ích khi sử dụng Railway.**

* Dễ dàng triển khai: Railway cung cấp một quy trình triển khai đơn giản và tự động từ kho lưu trữ, giúp giảm thiểu công sức và thời gian triển khai ứng dụng.
* Quản lý môi trường: Railway cho phép bạn tạo và quản lý các môi trường khác nhau cho ứng dụng của mình, giúp bạn kiểm thử, phát triển và triển khai ứng dụng một cách ddễ dàng.
* Tích hợp liền mạch: Railway tích hợp tốt với các kho lưu trữ phổ biến như GitHub và GitLab, giúp bạn tự động hóa quá trình triển khai từ mã nguồn.
* Quản lý biến môi trường: Railway cho phép bạn quản lý các biến môi trường, bao gồm cấu hình cơ sở dữ liệu, biến bí mật và các giá trị khác, giúp bảo mật và linh hoạt trong việc triển khai ứng dụng.
* Theo dõi và gỡ lỗi: Nền tảng Railway cung cấp giao diện để theo dõi tiến trình triển khai và xem các thông báo, nhật ký và lỗi. Điều này giúp bạn dễ dàng xác định vấn đề và sửa chữa nhanh chóng.

##### **2.1.3.4. Kết luận:**

- Railway là một nền tảng triển khai ứng dụng đám mây mạnh mẽ và tiện lợi. Qua báo cáo này, chúng tôi đã trình bày quy trình triển khai ứng dụng bằng Railway và các lợi ích của việc sử dụng nền tảng này. Bằng cách sử dụng Railway, bạn có thể triển khai ứng dụng một cách dễ dàng, tự động hóa và quản lý môi trường một cách hiệu quả.

### 2.2. Chức năng đăng ký, đăng nhập

#### **2.2.1. Giới thiệu Firebase**

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây - cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

Cụ thể là những giao diện lập trình ứng dụng API đơn giản. Mục đích nhằm tăng số lượng người dùng và thu lại nhiều lợi nhuận hơn.

Đặt biệt, còn là dịch vụ đa năng và bảo mật cực tốt. Firebase hỗ trợ của hai nền tảng Android và IOS. Không có gì khó hiểu khi nhiều lập trình viên chọn Firebase làm nền tảng đầu tiên để xây dựng ứng dụng cho hàng triệu người dùng trên toàn thế giới.

Ưu điểm sử dụng Firebase: Tạo tài khoản và sử dụng dễ dàng; Tốc độ phát triển nhanh; Nhiều dịch vụ trong một nền tảng; Được cung cấp bởi Google; Tập trung và phát triển giao diện người dùng; Firebase không có máy chủ,...

Các dịch vụ của Firebase có rất nhiều nhưng nhóm chỉ dùng 3 dịch vụ này cho việc xử lý dữ liệu đăng ký, đăng nhập:

* Authentication: là dịch vụ quản lý người dùng đơn giản và an toàn. Authentication cung cấp nhiều phương pháp xác thực email và mật khẩu Google, Facebook.
* Realtime Database: là dịch vụ lưu trữ và đồng bộ dữ liệu người dùng thời gian thực. Có hỗ trợ cho Android, IOS, Web, C++, Unity và Xamarin. Người dùng có thể lưu trữ và lấy dữ liệu từ máy chủ rất dễ dàng.
* Cloud Storage: là dịch vụ có khả năng lưu trữ và chia sẻ nội dung do người dùng tạo ra như hình ảnh, âm thanh và video với bộ nhớ mạnh, đơn giản và tiết kiệm chi phí được xây dựng cho quy mô của Google.

#### **2.2.2. Kết nối firebase**

Kết nối Firebase trong Android studio

* Mở Firebase trong Android Studio: Tool -> Firebase
* Kết nối Firebase Analytics
* Add Analytics vào app
* Add Firebase Authentication SDK vào app
* Add Realtime Database vào app
* Add Cloud Storage vào app

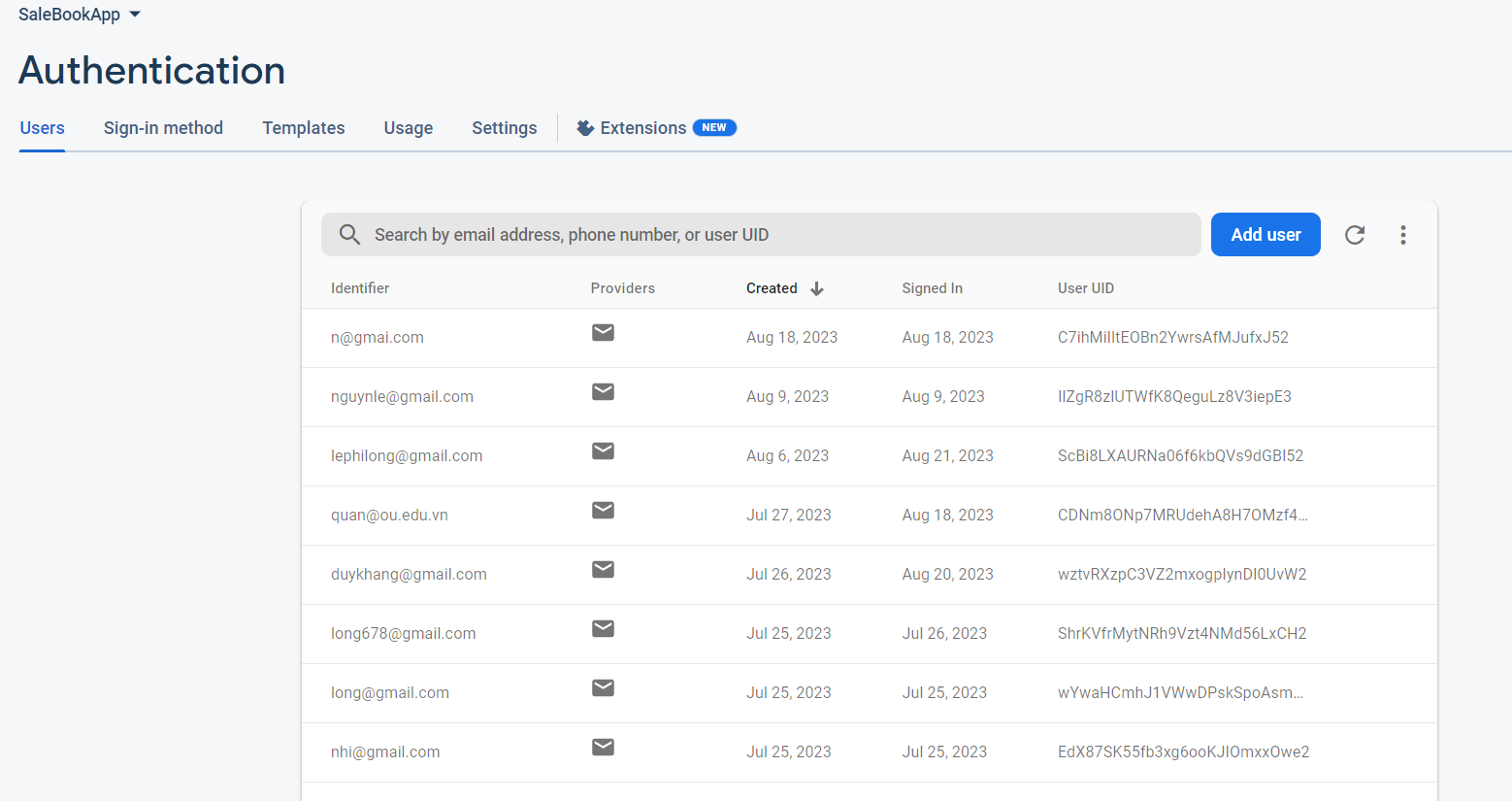
Trong build.gradle(Module:app) sẽ thêm các SDK này:

A screen shot of a computer

Description automatically generated

Kết nối Firebase trên Google:

Khởi động Authentication, Realtime Database, Storage bên Google Firebase đã được kết nối với dữ án. Sau là các giao diện của Firebase trên Google:



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

#### **2.2.3. Sử dụng firebase xử lý dữ liệu**

* 1. **Đăng ký:**
* Khai báo: private FirebaseAuth firebaseAuth Biến này dùng để tham chiếu đến đối tượng của lớp ‘ FirebaseAuth’, một phần của Firebase Authentication.
* Trong onCreate() phải khởi tạo Firebase auth để có thể sử dụng biến ‘firebaseAuth’: firebaseAuth= FirebaseAuth.getInstance();
* Khai báo: private ProgressDialog progressDialog; Biến này được sử dụng để tạo một hộp thoại tiến trình(ProgressDialog) để thông báo cho người dùng về tiến trình đang diễn ra trong ứng dụng. Trong bài sẽ hiển thị thông báo đang đợi khi thực hiện các tác vụ liên quan đến xác thực người dùng, như đăng nhập hoặc đăng ký. Điều này giúp người dùng biết rằng ứng dụng đang thực hiện một tác vụ nào đó và không bị treo hoặc không phản hồi.
* Trong onCreate() sẽ setup progress dialog:

A screen shot of a computer code

Description automatically generated

Đoạn code sẽ tạo ra một hộp thoại tiến trình có tiêu đề ‘please wait “ và ngăn người dùng đóng hộp thoại bằng cách chạm vào bên ngoài. Mục đích của việc này thường là để thông báo cho người dùng biết rằng ứng dụng đang thực hiện một thao tác nào đó và họ cần phải đợi cho đến khi tác vụ hoàn thành.

* Viết hàm tạo một tài khoản mới “ createAccount()”:

A screen shot of a computer program

Description automatically generated

Hàm này chứa logic để tạo tài khoản người dùng bằng Firebase Authentication. Nó đảm bảo rằng người dùng biết rằng ứng dụng đang thực hiện quá trình tạo tài khoản thông qua việc hiển thị hộp thoại tiến trình, và cung cấp phản hồi phù hợp cho người dùng. Quá trình được hoàn thành sẻ gọi phương thức “updateUserInfo()” cập nhật thông tin người dùng lên firebase. Quá trình thất bại thì hộp thoại sẽ được đóng bằng ‘progressDialog.dismiss()’ Một thông báo lỗi( dựa trên ngoại lệ ‘Exception’ được chuyển) sẽ được hiển thị cho người dùng thông qua ‘Toast’.

* Viết hàm cập nhật thông tin lên firebase ‘updateUserInfo()’:

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program

Description automatically generated

Hàm thực hiện việc lưu thông tin người dùng mới được tạo vào Firebase Realtime Database. Thông báo họp thoại tiến trình đang ‘Saving user info...’, tạo biến ‘timestamp’ để lấy thời gian hiện tại dưới dạng millisecond. Lấy ‘uid’ của người dùng từ ‘firebaseAuth.getUid()’ đây là cách bạn xác định người dùng hiện tại trong Firebase. Tạo đối tượng ‘HashMap’ để đóng gói thông tin của người dùng, tạo các thông tin bằng ‘HashMap.Put()’. Lấy tham chiếu đến node “Users” trên Firebase Realtime Database. Sử dụng ‘ref.child(uid).setValue(hashMap)’ để đặt giá trị của nút dưới ‘uid’ với dữ liệu trong ‘HashMap’. Nếu việc cập nhật thành công, phương thức ‘onSuccess’ được thực thi, đóng hộp thoại tiến trình báo thành công và chuyển người dùng đến màn hình đăng nhật. Ngược lại, phương thức ‘onFailure’ thực thi, đóng hộp thoại tiến trình và hiển thị thông báo lỗi cho người dùng.

* 1. **Đăng nhập:**
* Phần khai báo ‘private FirebaseAuth firebaseAuth’, ‘private ProgressDialog progressDialog;’ và init Firebase Auth, setup progress dialog tương tự như phần code ở đăng ký.
* Viết hàm đăng nhập người dùng ‘loginUser()’:

A screen shot of a computer code

Description automatically generated

Hàm thực hiện quy trình đăng nhật người dùng bằng Firebase Authentication. Nó đảm bảo rằng người dùng biết rằng ứng dụng đang thực hiện quá trình đăng nhập thông qua việc hiển thị hộp thoại tiến trình, và cung cấp phản hồi phù hợp cho người dùng. Sử dụng ‘firebaseAuth.signInWithEmailAndPassword(userName, passWord)’ đã khởi tạo trước thực hiện đăng nhập bằng cách sử dụng tên email và mật khẩu đã được đăng ký. Nếu đăng nhập thành công thì ‘onSuccess’ được thực thi, sẽ gọi phương thức ‘checkUser()’ để kiểm tra người dùng đã đăng ký thông tin cá nhân hay chưa. Ngược lại, phương thức ‘onFailure’ thực thi và đóng hộp thoại tiến trình, thông báo lỗi cho người dùng.

* Viết hàm kiểm tra có tài khoản của khách hàng trên Firebase chưa ‘checkUser()’:

A screen shot of a computer program

Description automatically generated

Hàm này thực thực hiện kiểm tra thông tin đã điền thông tin đăng nhập đã có tồn tại trong dữ liệu Firebase. Đặt thông báo cho hộp thoại tiến trình là“checking User…”.

Lấy đối tượng ‘FirebaseUser’ hiện tại thông qua ‘firebaseAuth.getCurrentUser()’ điều này cho phép bạn xác thực người dùng đang đăng nhập.

Lấy tham chiếu đến node ‘Users’ trên Firebase Realtime Database bằng cách sử dụng ‘FirebaseDatabase.getInstance().getReference("Users")’.

Sử dụng ‘firebaseUser.getUid()’ để lấy uid của người dùng hiện tại, điều này giúp bạn xác định nút tương ứng với thông tin của người dùng trong Firebase Realtime Database. Sử dụng ‘addListenerForSingleValueEvent’ để lắng nghe một lần duy nhất cho sự thay đổi dữ liệu tại nút người dùng tương ứng. Khi có dữ liệu thay đổi, phương thức ‘onDataChange’ được gọi, trong trường hợp này bạn lấy giá trị của trường “userType” từ ‘snapshot’ để xác thực loại người dùng. Lấy giá trị của trường “userType” từ dữ liệu ‘snapshot’. Nếu ‘userType’ là “user” có nghĩa là người dùng đăng nhập là người dùng bình thường, sau đó chuyển họ đến ‘MainActivity’ và kết thúc màn hình hiện tại. Phương thức ‘onCancelled’ được thực thi khi trường hợp xảy ra lỗi trong quá trình lắng nghe dữ liệu.

### 2.3. Chức năng hiển thị slide quảng cáo sản phẩm

#### **2.3.1. Giới thiệu thư viện glide**

Glide là một thư viện xử lý hình ảnh mạnh mẽ và linh hoạt được phát triển cho nền tảng Android. Thư viện này giúp quản lý và tải hình ảnh một cách hiệu quả trong ứng dụng Android của bạn. Dưới đây là một số điểm quan trọng về Glide:

1. **Tải hình ảnh hiệu quả**: Glide được thiết kế để tải hình ảnh một cách hiệu quả, giúp ứng dụng của bạn hoạt động mượt mà và tiết kiệm bộ nhớ.
2. **Caching**: Glide cung cấp hệ thống caching mạnh mẽ để lưu trữ hình ảnh đã tải. Điều này giúp giảm tải lên mạng và tăng tốc độ hiển thị hình ảnh khi người dùng xem lại chúng.
3. **Tải từ nhiều nguồn**: Glide cho phép tải hình ảnh từ nhiều nguồn khác nhau, bao gồm cả từ URL, tài nguyên cục bộ, và cả trong suốt từ dữ liệu bit.
4. **Tùy chỉnh và xử lý hình ảnh**: Glide cho phép bạn thực hiện nhiều xử lý hình ảnh như đổi kích thước, cắt, xoay và thậm chí áp dụng các bộ lọc hình ảnh trước khi hiển thị chúng.
5. **Tích hợp dễ dàng**: Glide tích hợp dễ dàng vào dự án Android của bạn bằng cách sử dụng Gradle hoặc Maven. Sau khi tích hợp, bạn có thể bắt đầu sử dụng nó ngay lập tức.
6. **Hỗ trợ GIF**: Glide hỗ trợ hiển thị GIF và cho phép bạn tải và xem các hình ảnh động này trong ứng dụng của bạn.
7. **Hỗ trợ đa luồng**: Glide hoạt động trên luồng nền, giúp tránh tình trạng treo ứng dụng khi tải hình ảnh trên luồng chính.
8. **Hỗ trợ đa dạng định dạng ảnh**: Glide hỗ trợ nhiều định dạng hình ảnh như JPEG, PNG, WebP, GIF và nhiều định dạng khác.

Sử dụng Glide làm cho việc quản lý và hiển thị hình ảnh trong ứng dụng Android trở nên dễ dàng và hiệu quả hơn, giúp bạn tập trung vào phát triển các tính năng chính của ứng dụng của mình mà không cần lo lắng về việc xử lý hình ảnh.



#### **2.3.2. Xử lý chức năng**

A screen shot of a computer program

Description automatically generated

1. **ActionViewFlipper()**: Đây là phương thức được định nghĩa bên trong một lớp hoặc fragment trong ứng dụng Android. Phương thức này sẽ thực hiện việc tạo và quản lý ViewFlipper để hiển thị quảng cáo.
2. **List<String> quangCao**: Đây là một danh sách chứa các URL hình ảnh quảng cáo. Các URL này sẽ được sử dụng để tải hình ảnh từ internet và hiển thị chúng trong ViewFlipper.
3. Trong vòng lặp **for**, một ImageView mới được tạo cho mỗi URL hình ảnh trong danh sách **quangCao**. Sau đó, thư viện Glide được sử dụng để tải hình ảnh từ URL và đặt nó vào ImageView. ImageView được cài đặt để thay đổi tỷ lệ và kích thước của hình ảnh để phù hợp với kích thước của ImageView.
4. **bannerAd**: Đây là một ViewFlipper, một thành phần giao diện người dùng trong Android được sử dụng để chuyển đổi giữa các giao diện con (trong trường hợp này là các hình ảnh quảng cáo). Các ImageView đã tạo ở bước trước được thêm vào ViewFlipper bằng cách sử dụng **bannerAd.addView(imageView)**.
5. **bannerAd.setFlipInterval(3000)**: Đặt khoảng thời gian giữa các chuyển đổi hình ảnh trong ViewFlipper là 3 giây (3000 milliseconds).
6. **bannerAd.setAutoStart(true)**: Thiết lập ViewFlipper tự động chuyển đổi giữa các hình ảnh khi ứng dụng khởi động.
7. **Animation slide\_in** và **Animation slide\_out**: Đây là hai animation (hiệu ứng) được định nghĩa trong tài liệu tài nguyên của ứng dụng (R.anim.slide\_in\_right và R.anim.slide\_out\_right). Các animation này được sử dụng để thực hiện hiệu ứng chuyển đổi khi ViewFlipper chuyển đổi giữa các hình ảnh. **setInAnimation** và **setOutAnimation** được sử dụng để đặt animation cho ViewFlipper.

Tóm lại, mã này tạo và quản lý một ViewFlipper trong ứng dụng Android, sử dụng Glide để tải và hiển thị các hình ảnh quảng cáo từ danh sách URL, và thiết lập các animation cho ViewFlipper để thực hiện hiệu ứng chuyển đổi giữa các hình ảnh.

A screen shot of a computer screen

Description automatically generated

* Đây là một tập hợp hoạt ảnh (animation set) trong Android.
* Hoạt ảnh bao gồm dịch chuyển (translation) từ vị trí ban đầu (-100%) đến vị trí cuối cùng (0%) theo chiều ngang và làm mờ (fade-in) từ độ trong suốt ban đầu (0.0) thành độ trong suốt cuối cùng (1.0).
* Thời gian thực hiện cả tập hợp hoạt ảnh này là 2 giây.

A screen shot of a computer program

Description automatically generated

* Đây là một tập hợp hoạt ảnh (animation set) trong Android.
* Hoạt ảnh bao gồm dịch chuyển (translation) từ vị trí ban đầu (0%) đến vị trí cuối cùng (100%) theo chiều ngang và làm mờ (fade-out) từ độ trong suốt ban đầu (1.0) thành độ trong suốt cuối cùng (0.0).
* Thời gian thực hiện cả tập hợp hoạt ảnh này là 2 giây.

Cả hai đoạn mã này có thể được sử dụng để tạo hiệu ứng chuyển động và làm mờ khi hiển thị hoặc ẩn đi một phần tử trong giao diện người dùng của ứng dụng Android.

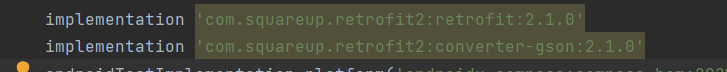
### 2.4. Chức năng hiển thị tác giả, sách, khuyến mãi

**2.4.1. Giới thiệu Retrofit**  
Retrofit là một thư viện mã nguồn mở phổ biến cho phát triển ứng dụng Android trong việc làm việc với các dịch vụ web RESTful. Nó được phát triển bởi Square, Inc. và cung cấp một cách tiện lợi và mạnh mẽ để thực hiện các yêu cầu HTTP và tương tác với API từ máy chủ từ xa. Dưới đây là một số điểm quan trọng về Retrofit:

1. **Sử dụng đơn giản**: Retrofit giúp đơn giản hóa việc giao tiếp với dịch vụ web RESTful. Nó sử dụng các ghi chú (annotations) để định nghĩa các yêu cầu HTTP và dựa vào các interface để mô tả cách ứng dụng của bạn sẽ tương tác với API.
2. **Tự động chuyển đổi JSON**: Retrofit tự động chuyển đổi dữ liệu JSON từ các yêu cầu và phản hồi thành các đối tượng Java, giúp bạn dễ dàng làm việc với dữ liệu từ máy chủ.
3. **Tùy chỉnh linh hoạt**: Bạn có thể tùy chỉnh Retrofit để thêm tiêu đề, thông tin xác thực, kiểm tra lỗi và nhiều tùy chọn khác cho các yêu cầu HTTP của bạn.
4. **Hỗ trợ Giao tiếp đa luồng**: Retrofit được tích hợp tốt với các thư viện khác như RxJava hoặc Kotlin Coroutines để thực hiện giao tiếp đa luồng một cách dễ dàng.
5. **Caching**: Retrofit hỗ trợ caching dựa trên tiêu đề Cache-Control trong các phản hồi từ máy chủ.
6. **Tự động thêm Log**: Nó cung cấp tích hợp sẵn cho việc ghi log cho các yêu cầu và phản hồi, giúp bạn theo dõi quá trình giao tiếp với máy chủ.
7. **Hỗ trợ các định dạng khác nhau**: Retrofit không chỉ hỗ trợ JSON mà còn hỗ trợ các định dạng dữ liệu khác nhau như XML, Protocol Buffers, và Scalars.
8. **Tiết kiệm thời gian và công sức**: Retrofit giúp tiết kiệm thời gian và công sức trong việc thực hiện các yêu cầu HTTP, giúp bạn tập trung vào phát triển các tính năng chính của ứng dụng của bạn.

Tóm lại, Retrofit là một thư viện phát triển Android mạnh mẽ và phổ biến cho việc giao tiếp với dịch vụ web RESTful. Nó giúp đơn giản hóa quá trình làm việc với API và tạo ra ứng dụng Android có khả năng tương tác với máy chủ từ xa một cách hiệu quả.

#### **2.4.2. Sử dụng Retrofit để gọi API trong Android Studio**



A screen shot of a computer program

Description automatically generated

1. **Gson gson = new GsonBuilder().setDateFormat("dd-MM-yyyy").create();**: Đoạn này tạo một đối tượng Gson, một công cụ giúp chuyển đổi dữ liệu JSON thành các đối tượng Java và ngược lại. Cài đặt **setDateFormat("dd-MM-yyyy")** được sử dụng để định dạng ngày tháng khi chuyển đổi.
2. **SachAPI apiInterface = (SachAPI) new Retrofit.Builder()...create(SachAPI.class);**: Đoạn này tạo một đối tượng Retrofit, một thư viện dùng để gửi các yêu cầu HTTP đến máy chủ từ xa và nhận phản hồi. Chi tiết:
   * **.baseUrl("https://book-api.up.railway.app/")**: Đây là địa chỉ gốc của máy chủ API, nơi mà các yêu cầu API sẽ được gửi đến.
   * **.addConverterFactory(GsonConverterFactory.create(gson))**: Retrofit cần biết cách chuyển đổi dữ liệu JSON thành các đối tượng Java và ngược lại. Chúng ta sử dụng GsonConverterFactory và cung cấp đối tượng Gson đã tạo.
   * **.build()**: Xây dựng một đối tượng Retrofit đã được cấu hình.
   * **.create(SachAPI.class)**: Tạo ra một đối tượng implement (thực hiện) interface SachAPI. Điều này cho phép sử dụng các phương thức trong interface để gửi yêu cầu API.
3. **@GET("api/book/getAll")**: Đây là một ghi chú (annotation) đánh dấu phương thức **getList()**, nó là yêu cầu HTTP loại GET và sẽ được gửi đến đường dẫn "api/book/getAll" trên máy chủ.
4. **Call<List<Sach>> getList();**: Đây là một phương thức trong interface SachAPI. Nó trả về một đối tượng **Call**, mà Retrofit sử dụng để thực hiện yêu cầu HTTP. Phương thức này dự kiến sẽ trả về một danh sách (List) các đối tượng Sach từ phản hồi của máy chủ từ xa.

Tóm lại, mã này định nghĩa một API interface sử dụng Retrofit để gửi yêu cầu HTTP đến máy chủ từ xa và nhận phản hồi. Các phản hồi từ máy chủ sau đó được chuyển đổi thành các đối tượng Java bằng Gson để dễ dàng xử lý trong ứng dụng Android.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

* **getListTacGia()** gửi một yêu cầu API để lấy danh sách Tác Giả.
* Trong trường hợp phản hồi thành công (**onResponse**), danh sách được lấy từ phản hồi và hiển thị bằng cách gắn vào một Adapter (tacGiaAdapter) và đặt vào RecyclerView (rcvTacGia).
* Trong trường hợp phản hồi thất bại (**onFailure**), hiển thị thông báo "Failure" bằng Toast.

A screen shot of a computer program

Description automatically generated

* **getListPopularBook()** gửi một yêu cầu API để lấy danh sách Sách Phổ Biến.
* Trong trường hợp phản hồi thành công (**onResponse**), danh sách được lấy từ phản hồi và hiển thị bằng cách gắn vào một Adapter (bookAdapter) và đặt vào GridView (gvPopularBook).
* Trong trường hợp phản hồi thất bại (**onFailure**), hiển thị thông báo "Failure" bằng Toast.

Cả hai phương thức này có cùng cấu trúc và mục tiêu là lấy dữ liệu từ API và hiển thị nó trong giao diện người dùng. Phần quan trọng là sử dụng Retrofit để thực hiện yêu cầu và xử lý phản hồi một cách đáng tin cậy.

### 2.5. Chức năng tìm kiếm sản phẩm

#### **2.5.1. Xử lý chức năng**

- Trong SachAdapter

A screen shot of a computer program

Description automatically generated

- Trong SearchActivity

A computer screen shot of a program code

Description automatically generated

* Phương thức này nhận vào một chuỗi văn bản **text** để tìm kiếm trong danh sách **listSach**.
* Nó tạo một danh sách trống **filteredlist** để lưu trữ các mục sách được tìm thấy.
* Sau đó, nó duyệt qua từng phần tử **item** trong danh sách **listSach**.
* Đối với mỗi **item**, nó kiểm tra xem chuỗi **text** có xuất hiện trong tiêu đề (**getTitle()**) của **item** không (bằng cách so sánh chuỗi với biến đối thước **toLowerCase()**).
* Nếu **text** xuất hiện trong tiêu đề của **item**, nó thêm **item** đó vào danh sách **filteredlist**.
* Sau khi duyệt xong, nếu **filteredlist** là trống, nghĩa là không có kết quả tìm kiếm, thì hiển thị thông báo "No Data Found". Ngược lại, nó gọi phương thức **filterList()** của adapter để cập nhật dữ liệu hiển thị trên GridView (**gvBook**) bằng danh sách **filteredlist**.

A computer screen shot of text

Description automatically generated

1. **@Override public boolean onCreateOptionsMenu(Menu menu)**: Phương thức này được ghi đè để tạo menu cho hoạt động (activity) của bạn. Nó nhận tham số là một đối tượng **Menu** và trả về **true** để hiển thị menu hoặc **false** để ẩn nó.
2. **MenuInflater inflater = getMenuInflater();**: Dùng để lấy một đối tượng **MenuInflater**, giúp bạn "nạp" (inflate) các menu từ tài nguyên.
3. **inflater.inflate(R.menu.search\_menu, menu);**: Đoạn này inflate (nạp) menu từ tài nguyên với ID **search\_menu** và đưa vào menu của hoạt động (**menu**).
4. **MenuItem searchItem = menu.findItem(R.id.actionSearch);**: Tìm kiếm một mục menu với ID **actionSearch** trong menu đã được inflate.
5. **SearchView searchView = (SearchView) searchItem.getActionView();**: Lấy thanh tìm kiếm từ mục menu đã tìm thấy. Điều này cho phép tương tác với thanh tìm kiếm.
6. **searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() { ... }**: Trong trường hợp này, nó sử dụng **OnQueryTextListener** để thực hiện sự kiện khi người dùng nhập hoặc thay đổi văn bản trong thanh tìm kiếm.
   * **onQueryTextSubmit(String query)**: Phương thức này được gọi khi người dùng nhấn Enter hoặc nút tìm kiếm trên bàn phím sau khi nhập văn bản. Trong trường hợp này, nó không thực hiện hành động gì và trả về **false** để chỉ định rằng sự kiện đã được xử lý.
   * **onQueryTextChange(String newText)**: Phương thức này được gọi khi người dùng thay đổi nội dung trong thanh tìm kiếm. Trong trường hợp này, nó gọi phương thức **filter(newText)** để tìm kiếm và lọc dữ liệu dựa trên văn bản mới nhập vào thanh tìm kiếm và trả về **false** để chỉ định rằng sự kiện đã được xử lý.

Tổng cộng, đoạn mã này cho phép bạn thêm một thanh tìm kiếm vào menu của hoạt động Android và sự kiện tìm kiếm để thực hiện các tác vụ tìm kiếm dựa trên dữ liệu trong ứng dụng.

### 2.6. Chức năng sắp xếp sách theo thể loại trong danh mục

#### **2.6.1. Xử lý chức năng**

A screen shot of a computer program

Description automatically generated

1. **Gson Configuration**:
   * **Gson gson = new GsonBuilder().setDateFormat("dd-MM-yyyy").create();**: Đoạn này tạo một đối tượng Gson để chuyển đổi dữ liệu JSON thành các đối tượng Java và ngược lại. Đặc điểm quan trọng ở đây là định dạng ngày tháng được cấu hình là "dd-MM-yyyy".
2. **Tạo Retrofit API Interface**:
   * **DanhMucAPI apiInterface = new Retrofit.Builder()...create(DanhMucAPI.class);**: Đoạn này tạo một đối tượng Retrofit để tương tác với dịch vụ web từ xa. Cụ thể:
     + **.baseUrl("https://book-api.up.railway.app/")**: Địa chỉ gốc của máy chủ API, nơi mà các yêu cầu API sẽ được gửi đến.
     + **.addConverterFactory(GsonConverterFactory.create(gson))**: Retrofit cần biết cách chuyển đổi dữ liệu JSON thành các đối tượng Java và ngược lại. Ở đây, chúng ta sử dụng GsonConverterFactory và cung cấp đối tượng Gson đã được tạo ở trên.
     + **.build()**: Xây dựng một đối tượng Retrofit đã được cấu hình.
     + **.create(DanhMucAPI.class)**: Tạo một đối tượng implement (thực hiện) interface **DanhMucAPI**. Điều này cho phép sử dụng các phương thức trong interface này để gửi yêu cầu API.
3. **Các phương thức API**:
   * Các phương thức được định nghĩa trong giao diện này đại diện cho các yêu cầu API cụ thể liên quan đến các danh mục sách khác nhau. Ví dụ:
     + **@GET("api/book/getByCategory/3222")**: Đây là một phương thức để lấy sách thuộc danh mục "Thiếu Nhi". Mã số 3222 có thể đại diện cho mã danh mục của sách "Thiếu Nhi" trên máy chủ API.
     + Tương tự, các phương thức khác đại diện cho các danh mục sách khác nhau như "Kinh Doanh", "Khoa Học", "Văn Hóa", "Lịch Sử", và "Giáo Dục".

Mã này cho phép tạo các yêu cầu API dễ dàng thông qua các phương thức đã được định nghĩa trong giao diện **DanhMucAPI**, và sử dụng Retrofit để thực hiện các yêu cầu này và nhận phản hồi từ máy chủ từ xa dưới dạng danh sách sách thuộc các danh mục khác nhau.

- Trong CategoryFragment

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

- Trên đây là cách gọi ra 2 API của 2 loại thể loại sách, các thể loại khác tương tự.

1. **context = CategoryFragment.this.getContext();**: Đoạn này gán biến **context** bằng **getContext()** của Fragment hiện tại. Biến **context** sẽ được sử dụng sau này để hiển thị Toast và để khởi tạo Intent.
2. **View view = inflater.inflate(R.layout.fragment\_category, container, false);**: Đoạn này inflate (nạp) giao diện người dùng từ tệp **fragment\_category.xml** vào biến **view**. Đây là giao diện cho Fragment này.
3. **gvBook = view.findViewById(R.id.gv\_sach);**: Tìm và lưu trữ tham chiếu đến GridView có ID **gv\_sach** trong giao diện người dùng. GridView này dùng để hiển thị danh sách sách.
4. **Toast.makeText(context, "Giáo dục", Toast.LENGTH\_SHORT).show();**: Hiển thị một thông báo ngắn thông qua Toast để thể hiện danh mục "Giáo dục".
5. **getGiaoDuc();**: Gọi phương thức **getGiaoDuc()** để lấy danh sách sách thuộc danh mục "Giáo dục" và hiển thị chúng trong GridView.
6. Các đoạn mã từ **tvSearch = view.findViewById(R.id.tv\_timkiem);** đến **lnrLichSu = view.findViewById(R.id.lnr\_lichsu);** thực hiện việc tìm và lưu trữ tham chiếu đến các phần tử giao diện người dùng khác nhau. Các phần tử này có thể là nút hoặc layout để người dùng có thể tương tác và chọn danh mục sách khác nhau.
7. Mỗi phần tử giao diện được kết hợp với một sự kiện **OnClickListener**, để khi người dùng nhấp vào chúng, các phương thức tương ứng (**getGiaoDuc()**, **getThieuNhi()**, vv.) sẽ được gọi để lấy danh sách sách tương ứng và hiển thị chúng trong GridView. Ngoài ra, Toast cũng được sử dụng để hiển thị tên danh mục được chọn.
8. Cuối cùng, phương thức **onCreateView** trả về biến **view**, đại diện cho giao diện người dùng của Fragment này. Giao diện này chứa GridView và các phần tử tương tác khác để người dùng có thể xem sách trong các danh mục khác nhau.

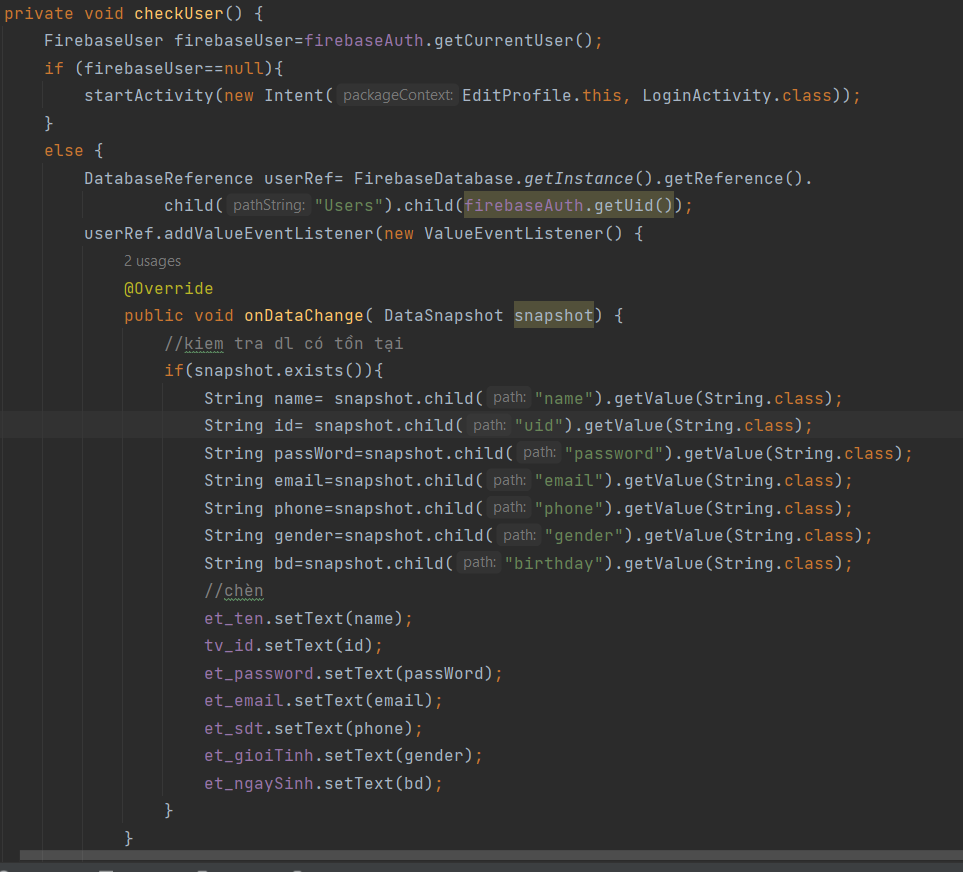
### 2.7. Chức năng tùy chỉnh thông tin hồ sơ cá nhân

#### **2.7.1. Xử lý chức năng**

- Khai báo: private FirebaseAuth firebaseAuth Biến này dùng để tham chiếu đến đối tượng của lớp ‘ FirebaseAuth’, một phần của Firebase Authentication.

- Trong onCreate() phải khởi tạo Firebase auth để có thể sử dụng biến ‘firebaseAuth’: firebaseAuth= FirebaseAuth.getInstance();

- Viết hàm kiểm tra tài khoản khách hàng “ checkUser”:



Hàm sử dụng Firebase Authentication và Firebase Realtime Database trong ứng dụng Android để kiểm tra tình trạng đăng nhập của người dùng và hiển thị thông tin cá nhân của họ trong màn hình chỉnh sửa thông tin (EditProfile).

‘FirebaseUser firebaseUser=firebaseAuth.getCurrentUser();’ : Đoạn mã này lấy thông tin người dùng hiện tại đang đăng nhập từ Firebase Authentication.

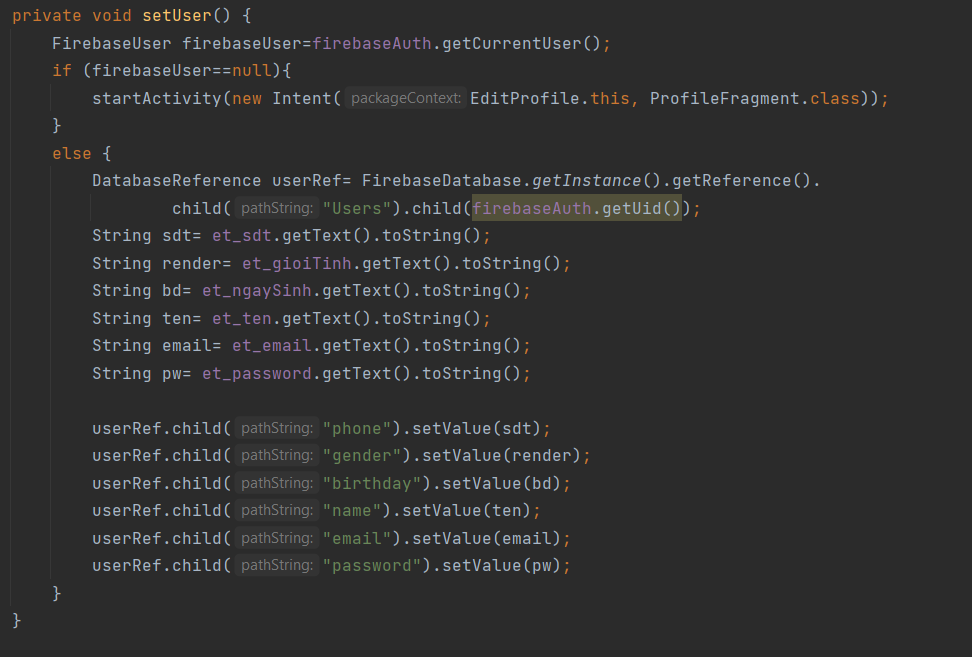
‘startActivity(new Intent(EditProfile.this, LoginActivity.class));’: Nếu người dùng chưa đăng nhập, màn hình sẽ chuyển hướng đến màn hình đăng nhập (LoginActivity).

‘DatabaseReference\_userRef=FirebaseDatabase.getInstance().getReference().child("Users").child(firebaseAuth.getUid());’: Đoạn mã này xây dựng một đối tượng DatabaseReference để truy cập vào nút "Users" trong Firebase Realtime Database và lấy dữ liệu của người dùng dựa trên UID của người dùng đang đăng nhập.

‘public void onDataChange( DataSnapshot snapshot) {...}’: Phương thức này được gọi khi dữ liệu thay đổi trên userRef. Trong phương thức này, bạn kiểm tra xem liệu dữ liệu người dùng có tồn tại trong cơ sở dữ liệu hay không thông qua snapshot.exists().

Lấy thông tin từ ‘snapshot’: Trong trường hợp dữ liệu người dùng tồn tại, bạn trích xuất thông tin như tên, ID, mật khẩu, email, số điện thoại, giới tính và ngày sinh từ snapshot và hiển thị chúng trên các trường nhập liệu (EditText) hoặc các thành phần giao diện khác.

- Viết hàm cập nhật thông tin lên Firebase ‘setUser()’:



FirebaseUser firebaseUser=firebaseAuth.getCurrentUser();: Đoạn mã này lấy thông tin người dùng hiện tại đang đăng nhập từ Firebase Authentication.

Kiểm tra tình trạng đăng nhập của người dùng: Nếu người dùng chưa đăng nhập (biến firebaseUser là null), mã nguồn sẽ chuyển hướng người dùng đến màn hình hồ sơ cá nhân (ProfileFragment). Tuy nhiên, trong trường hợp này, có vẻ hơi kỳ quặc để chuyển hướng đến một Fragment thay vì một Activity. Thường thì chúng ta sẽ chuyển hướng đến một Activity để đăng nhập.

Lấy thông tin từ các trường nhập liệu: Đoạn mã này lấy thông tin từ các trường nhập liệu (như số điện thoại, giới tính, ngày sinh, tên, email, mật khẩu) mà người dùng đã điền để cập nhật thông tin cá nhân.

DatabaseReference\_userRef=FirebaseDatabase.getInstance().getReference().child("Users").child(firebaseAuth.getUid());: Đoạn mã này xây dựng một đối tượng DatabaseReference để truy cập vào nút "Users" trong Firebase Realtime Database và lấy dữ liệu của người dùng dựa trên UID của người dùng đang đăng nhập.

Cập nhật thông tin vào Firebase Realtime Database: Sử dụng các child() để xác định nút con cần cập nhật, ví dụ như "phone", "gender", "birthday", v.v. Sau đó, bạn sử dụng setValue() để gán giá trị mới vào các nút con này, cập nhật thông tin cá nhân của người dùng.

### 2.8. Chức năng thêm sách vào giỏ hàng

#### **2.8.1. Xử lý chức năng**

Sử dụng setOnClickListener trong SachAdapder để bắt sự kiện nhấn nút “Mua” của từng sản phẩm.

A screen shot of a computer

Description automatically generated

Khởi tạo một Intent để lấy dữ liệu và dùng putExtra của Intent để lấy dữ liệu(hình ảnh, tiêu đề, giá cả) của một cuốn sách bất kì khi click vào.

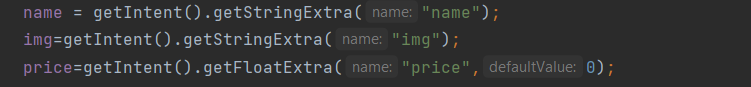
Gọi StartActivity để đẩy dữ liệu đi.

A screen shot of a computer code

Description automatically generated

Sau đó tạo activity\_add\_cart( Giỏ hàng) để hứng dữ liệu xuống.

Sử dụng getIntent().getStringExtra() để hứng các dữ liệu kiểu chuỗi(gồm tiêu đề và hình ảnh) và getIntent().getFloatExtra() để hứng các dữ liệu kiểu số thực(giá).



Tạo 1 Adapter CartAdapter tạo các sản phẩm trong giỏ hàng với dữ liệu vừa được lấy xuống.

Cuối cùng add Adapter vào giao diện danh sách được tạo sẵn(GridView) để bỏ sản phẩm vào.



Ngoài ra trên trang Cart còn có cái nút như Thanh Toán và xoá trên mỗi sản phẩm sử dụng setOnClickListener và dùng list.remove() và list.removeAll() để xoá khỏi giỏ hàng.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

## 3. **Hướng dẫn sử dụng:**

### 3.1. Đăng ký, đăng nhập

Khởi động ứng dụng bán sách trên thiết bị di động của bạn.

Ứng dụng sẽ hiện màn hình đăng nhập, chọn tùy chọn “Đăng ký”. Nhập thông tin cá nhân cần thiết như tên đăng nhập, địa chỉ email, mật khẩu lần 1, mật khẩu lần 2.

Xác nhận lại thông tin bạn vừa điền để đảm bảo tính chính xác. Hoàn tất đăng ký bấm nút “Đăng ký” để hoàn tất quá trình.

Ứng dụng sẽ hiển thị lại màn hình đăng nhập, người dùng nhập thông tin đăng nhập bao gồm: địa chỉ email và mật khẩu đã đăng ký.

Bấm vào nút “Đăng nhập” để xác thực thông tin và truy cập vào tài khoản cá nhân.

### 3.2. Chỉnh sửa hồ sơ

Vào ứng dụng tìm và chọn biểu tượng giao diện hồ sơ cá nhân “Profile”. Nhấn chọn “Edit” để bắt đầu chỉnh sửa hồ sơ cá nhân.

Xem thông tin cá nhân của mình như tên, email, mật khẩu, số điện thoại, ngày sinh. Nếu có thông tin chưa chính xác hay muốn thay đổi thông tin khác thì khách hàng điền hoặc chỉnh sửa thông tin cần thay đổi.

Sau khi đã chỉnh sửa thông tin, bấm vào nút “Save” để cập nhật thông tin cá nhân mới.

### 3.3. Chọn Sản phẩm và Thêm vào Giỏ hàng

Khách hàng duyệt qua trang web bán sách và tìm kiếm sách mình muốn mua bằng cách sử dụng chức năng tìm kiếm hoặc duyệt qua các danh mục.

Khi khách hàng tìm thấy sản phẩm mong muốn, họ có thể xem qua hình ảnh và chi tiết thông tin sách để đảm bảo đúng sách mà họ muốn.

Mỗi sản phẩm đều có chi tiết giá cả hình ảnh và tiêu đề để khách hàng tham khảo và lựa chọn.

Khách hàng có thể chọn sách mà họ muốn mua, sau đó nhấp vào nút "Mua" để thêm sản phẩm vào giỏ hàng.

Hệ thống sẽ xác nhận việc thêm sản phẩm vào giỏ hàng và hiển thị một thông báo xác nhận.

### 3.4. Quản lý giỏ hàng

Sau khi thêm sản phẩm vào giỏ hàng, khách hàng có thể nhấp vào biểu tượng giỏ hàng kế biểu tưởng danh mục để xem thông tin chi tiết về giỏ hàng của mình.

Trong trang giỏ hàng, khách hàng có thể xem tất cả các sản phẩm đã thêm, hình ảnh, giá và tổng tiền của từng sản phẩm.

Họ cũng có thể cập nhật giỏ hàng, hoặc xóa sản phẩm ra khỏi giỏ hàng nếu cần.

### 3.5. Thanh toán

Sau khi quản lý giỏ hàng và chắc chắn muốn mua các sản phẩm đã chọn, khách hàng có thể nhấp vào nút "Thanh toán" để thanh toán.

Họ cũng có thể xem lại đơn hàng và tổng số tiền trước khi xác nhận thanh toán.

Sau khi xác nhận, hệ thống sẽ xử lý đơn hàng, trừ tiền từ tài khoản của khách hàng hoặc thông qua phương thức thanh toán được chọn.