

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ VII, NĂM HỌC 2023 - 2024

Nghiên cứu sử dụng RESTful API và Flutter để xây dựng ứng dụng di động quản lý quán Cafe

Giáo viên hướng dẫn:
ThS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Nhật Sang
MSSV: 110120151
Lớp: DA20TTB

Trà Vinh, tháng 1 năm 2024

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH
HỌC KỲ VII, NĂM HỌC 2023 - 2024**

**Nghiên cứu sử dụng RESTful API và
Flutter để xây dựng ứng dụng di động
quản lý quán Cafe**

Giáo viên hướng dẫn:
ThS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Nhất Sang
MSSV: 110120151
Lớp: DA20TTB

Trà Vinh, tháng 1 năm 2024

[illegible]

Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

[illegible]

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trên hết, em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Bảo Ân, giảng viên Khoa Kỹ thuật & Công nghệ, người đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình làm đồ án chuyên ngành.

Em cũng xin chân thành cảm ơn các Thầy (Cô) trong trường Đại học Trà Vinh nói chung, các Thầy (Cô) trong Khoa Kỹ thuật & Công nghệ nói riêng đã dạy dỗ cho em kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em trong suốt quá trình học tập.

Cuối cùng, trong quá trình tìm hiểu và thực hiện đề tài đồ án chuyên ngành, em vẫn còn những hạn chế và gặp những khó khăn. Vì thế không tránh khỏi những sai sót, kính mong nhận được những ý kiến đóng góp từ những thầy cô.

Trà Vinh, ngày tháng năm 2023

Sinh viên thực hiện

MỤC LỤC

| | |
|--|-----------|
| DANH MỤC HÌNH ẢNH | 9 |
| DANH MỤC BẢNG BIỂU | 10 |
| DANH MỤC TỪ VIẾT TẮT..... | 11 |
| TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH | 12 |
| MỞ ĐẦU | 13 |
| CHƯƠNG 1. TỔNG QUAN..... | 14 |
| CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT | 15 |
| 2. 1 Flutter | 15 |
| 2. 1. 1 Giới thiệu chung về Flutter..... | 15 |
| 2. 1. 2 Kiến trúc cơ bản của Flutter | 16 |
| 2. 1. 2. 1 Các thành phần chính..... | 16 |
| 2. 1. 2. 2 Các thành phần framework | 17 |
| 2. 1. 2. 3 Mô hình hoạt động của Flutter..... | 18 |
| 2. 1. 3 Ngôn ngữ lập trình Flutter | 19 |
| 2. 1. 3. 1 Ngôn ngữ Dart..... | 19 |
| 2. 1. 3. 2 Cấu trúc thư mục trong dự án Flutter | 20 |
| 2. 1. 3. 3 Các tính năng nổi bật của Dart | 21 |
| 2. 1. 3. 4 Dart hỗ trợ phát triển ứng dụng với Flutter | 21 |
| 2. 1. 4 Quản lý trạng thái ứng dụng | 22 |
| 2. 1. 5 Ưu nhược điểm Flutter..... | 22 |
| 2. 1. 5. 1 Ưu điểm Flutter | 22 |
| 2. 1. 5. 2 Nhược điểm Flutter..... | 23 |
| 2. 2 RESTful API | 23 |
| 2. 2. 1 Khái niệm REST và RESTful API | 23 |
| 2. 2. 1. 1 Giới thiệu về kiến trúc REST | 23 |
| 2. 2. 1. 2 Định nghĩa RESTful API..... | 23 |
| 2. 2. 1. 3 Các ràng buộc trong RESTful API | 24 |
| 2. 2. 2 Các thành phần của RESTful API | 24 |
| 2. 2. 2. 1 Resource | 24 |
| 2. 2. 2. 2 URI..... | 24 |
| 2. 2. 2. 3 Representation | 25 |
| 2. 2. 2. 4 Method..... | 25 |
| 2. 2. 3 Các status code trong RESTful API | 25 |
| 2. 2. 4 Xác thực API | 25 |
| 2. 2. 4. 1 Tầm quan trọng của việc xác thực trong RESTful API..... | 25 |
| 2. 2. 4. 2 Các phương thức xác thực API phổ biến..... | 25 |

| | |
|--|-----------|
| 2. 2. 4. 3 Lợi ích của việc xác thực API..... | 26 |
| CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU | 27 |
| 3. 1 Đặt tả ứng dụng | 27 |
| 3. 1. 1 Mô hình dữ liệu mức quan hệ..... | 27 |
| 3. 1. 2 Mô hình dữ liệu mức vật lý | 27 |
| 3. 1. 3 Sơ đồ Use-case..... | 29 |
| 3. 1. 3. 1 Sơ đồ | 29 |
| 3. 1. 3. 2 Danh sách Actor | 29 |
| 3. 1. 3. 3 Danh sách Use-case | 30 |
| 3. 1. 4 Sơ đồ hoạt động | 30 |
| 3. 1. 5 Sơ đồ tuần tự..... | 31 |
| 3. 2 Kiến trúc hệ thống | 32 |
| 3. 2. 1 Client (Flutter) | 32 |
| 3. 2. 2 Firebase Backend..... | 33 |
| 3. 3 Thiết kế RESTful API | 33 |
| 3. 3. 1 Đọc hóa đơn..... | 33 |
| 3. 3. 2 Đọc hóa đơn theo Id..... | 33 |
| 3. 3. 3 Tạo hóa đơn mới | 34 |
| 3. 3. 4 Cập nhật hóa đơn | 34 |
| 3. 3. 5 Xóa hóa đơn..... | 35 |
| 3. 4 Tích hợp Flutter với RESTful API | 35 |
| CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU | 38 |
| 4. 1 Giao diện..... | 38 |
| 4. 1. 1 Giao diện đăng nhập | 38 |
| 4. 1. 2 Giao diện danh sách bàn | 39 |
| 4. 1. 3 Giao diện danh sách sản phẩm..... | 40 |
| 4. 1. 4 Giao diện chi tiết order | 41 |
| 4. 2 Chức năng..... | 42 |
| 4. 2. 1 Quản lý bàn..... | 42 |
| 4. 2. 2 Quản lý danh sách sản phẩm | 43 |
| 4. 2. 3 Chi tiết order | 44 |
| CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN..... | 45 |
| 5. 1 Kết luận | 45 |
| 5. 1. 1 Kết quả đạt được | 45 |
| 5. 1. 2 Đóng góp mới | 45 |
| 5. 1. 3 Đề xuất mới | 45 |
| 5. 2 Hướng phát triển: | 45 |

| | |
|--|-----------|
| DANH MỤC TÀI LIỆU THAM KHẢO | 46 |
|--|-----------|

DANH MỤC HÌNH ẢNH

| | |
|--|----|
| Hình 1. Flutter..... | 15 |
| Hình 2. Widgets | 17 |
| Hình 3. Layers | 18 |
| Hình 4. Ngôn ngữ Dart | 19 |
| Hình 5. Cấu trúc thư mục trong dự án Flutter | 20 |
| Hình 6. Mô hình dữ liệu mức quan hệ..... | 27 |
| Hình 7. Mô hình dữ liệu mức vật lý | 27 |
| Hình 8. Sơ đồ Use-case..... | 29 |
| Hình 9. Sơ đồ order đồ uống..... | 30 |
| Hình 10. Sơ đồ tuần tự order đồ uống | 31 |
| Hình 11. Đọc hóa đơn..... | 33 |
| Hình 12. Đọc hóa đơn theo Id..... | 34 |
| Hình 13. Tạo hóa đơn mới..... | 34 |
| Hình 14. Cập nhật hóa đơn | 34 |
| Hình 15. Xóa hóa đơn..... | 35 |
| Hình 16. Gửi request GET | 36 |
| Hình 17. Gửi request POST | 36 |
| Hình 18. Hiển thị dữ liệu | 37 |
| Hình 19. Sử dụng ListView.builder để hiển thị dữ liệu..... | 37 |
| Hình 20. Xử lý mã trạng thái | 37 |
| Hình 21. Giao diện đăng nhập | 38 |
| Hình 22. Giao diện danh sách bàn | 39 |
| Hình 23. Giao diện danh sách sản phẩm..... | 40 |
| Hình 24. Giao diện chi tiết order | 41 |
| Hình 25. Hiển thị danh sách bàn..... | 42 |
| Hình 26. Hiển thị danh sách sản phẩm | 43 |
| Hình 27. Chi tiết order | 44 |

DANH MỤC BẢNG BIỂU

| | |
|--|----|
| Bảng 1. Bảng danh mục từ viết tắt..... | 11 |
| Bảng 2. Bảng user..... | 28 |
| Bảng 3. Bảng table..... | 28 |
| Bảng 4. Bảng invoice | 28 |
| Bảng 5. Bảng category..... | 28 |
| Bảng 6. Bảng product | 29 |
| Bảng 7. Bảng danh sách Actor..... | 29 |
| Bảng 8. Bảng danh sách Use-case | 30 |

DANH MỤC TỪ VIẾT TẮT

Bảng 1. Bảng danh mục từ viết tắt

| Từ viết tắt | Nghĩa của từ viết tắt |
|--------------------|-----------------------------------|
| AOT | Ahead of Time |
| API | Application Programming Interface |
| CSDL | Cơ sở dữ liệu |
| HTML | HyperText Markup Language |
| HTTP | Hypertext Transfer Protocol |
| JIT | Just-In-Time |
| JSON | JavaScript Object Notation |
| OOP | Object-Oriented Programming |
| REST | Representational State Transfer |
| SDK | Software Development Kit |
| UI | User Interface |
| URI | Uniform Resource Identifier |
| VM | Virtual Machine |
| XML | Extensible Markup Language |

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Trong đồ án, tôi đã vận dụng kiến thức đã được học về Flutter và RESTful API để xây dựng phần mềm quản lý hoạt động cho quán cafe trên nền tảng Android và IOS. Cụ thể, Flutter được dùng để xây dựng giao diện người dùng trực quan, dễ sử dụng. RESTful API đảm nhiệm vai trò kết nối và quản lý dữ liệu giữa phần mềm và cơ sở dữ liệu. Kết quả, tôi đã hoàn thành bản demo phần mềm quản lý quán cafe với các chức năng cơ bản như: quản lý bàn, gọi đồ uống, quản lý đơn hàng, thanh toán và thống kê báo cáo.

MỞ ĐẦU

Trong bối cảnh công nghệ thông tin phát triển vượt bậc, việc ứng dụng công nghệ vào quản lý kinh doanh dịch vụ, đặc biệt là các dịch vụ ăn uống như nhà hàng, quán cafe là vô cùng cần thiết. Các hệ thống quản lý tự động sẽ giúp tiết kiệm thời gian, công sức và tài nguyên cho chủ kinh doanh. Vì vậy, em chọn đề tài “Nghiên cứu sử dụng RESTful API và Flutter để xây dựng ứng dụng di động quản lý quán Cafe” với mong muốn đem đến một giải pháp công nghệ hiệu quả cho các chủ quán cafe.

Mục tiêu của đề tài là nghiên cứu và xây dựng phần mềm quản lý hoạt động cho các quán cafe trên nền tảng di động, cụ thể là hệ điều hành Android và iOS. Đối tượng nghiên cứu chính là các quán cafe có nhu cầu trong việc cải thiện quy trình quản lý bằng công nghệ.

Phạm vi nghiên cứu bao gồm các công nghệ và ngôn ngữ lập trình để xây dựng phần mềm quản lý gồm Flutter cho phần front-end và RESTful API cho phần back-end. Nghiên cứu sẽ tập trung vào việc thiết kế giao diện người dùng sao cho phù hợp và dễ sử dụng đối với nhân viên và quản lý quán cafe. Bên cạnh đó, việc tích hợp các tính năng như quản lý đơn hàng, theo dõi doanh thu, quản lý nhân sự và tương tác với khách hàng cũng sẽ được khám phá.

CHƯƠNG 1. TỔNG QUAN

Quá trình quản lý khách hàng và các hoạt động của quán cafe thường được thực hiện bằng cách ghi chép thủ công, gây mất thời gian và khó kiểm soát. Do vậy, cần một hệ thống quản lý quán cafe tự động hóa để đơn giản hóa khâu quản lý và nâng cao hiệu quả hoạt động của quán. Tôi quyết định sử dụng RESTful API và Flutter như các công cụ xây dựng ứng dụng quản lý quán cafe trên mobile.

RESTful API là tiêu chuẩn thiết kế Web API, cho phép tương tác với cơ sở dữ liệu thông qua các giao thức HTTP. Flutter là framework phát triển ứng dụng mobile đa nền tảng. Cả hai giúp xây dựng ứng dụng quản lý quán cafe dễ dàng và tối ưu trên nhiều thiết bị.

Sau quá trình nghiên cứu và thử nghiệm, tôi đã sử dụng RESTful API để xây dựng back-end và Flutter để xây dựng front-end cho ứng dụng quản lý quán cafe trên điện thoại di động. Ứng dụng cho phép quản lý các hoạt động như đặt bàn, gọi món, thanh toán một cách hiệu quả.

CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT

2. 1 Flutter

2. 1. 1 Giới thiệu chung về Flutter



Hình 1. Flutter

Flutter là một framework mã nguồn mở do Google phát triển. Nó cho phép nhà phát triển xây dựng ứng dụng đẹp mắt, nhanh chóng và hiệu quả bằng cách sử dụng một mã nguồn duy nhất. Hỗ trợ phát triển ứng dụng đa nền tảng như Mobile, Web, Desktop, Embedded [1]. Theo định nghĩa của Google trên trang chính thức của Flutter, được mô tả như sau:

“Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.” – Google, flutter.dev.

Flutter không chỉ là một framework hay thư viện, nó là một Software Development Kit (SDK) đầy đủ, hỗ trợ phát triển đa nền tảng. Flutter cho phép phát triển ứng dụng di động trên cả hai nền tảng iOS và Android chỉ với một lần viết mã nguồn và sử dụng một ngôn ngữ lập trình duy nhất.

Flutter sử dụng ngôn ngữ lập trình Dart, được biết đến với hiệu suất cao, độ ổn định và tính dễ học. Đặc biệt, Flutter nổi bật với khả năng xây dựng giao diện người dùng thông qua sự kết hợp linh hoạt của các thành phần UI tùy chỉnh.

Cung cấp nhiều công cụ, thư viện hỗ trợ. Trong quá trình phát triển ứng dụng di động, tính năng Hot Reload là điểm cộng trong Flutter, giúp các lập trình viên thấy ngay các thay đổi mà không cần khởi động lại toàn bộ ứng dụng, từ đó giảm thời gian và công sức.

Tóm lại, Flutter đại diện cho một framework mã nguồn mở tiên tiến của Google, là một giải pháp lý tưởng cho việc phát triển ứng dụng di động chất lượng cao, đa nền tảng, nhờ việc kết hợp một codebase và một ngôn ngữ lập trình duy nhất.

2. 1. 2 Kiến trúc cơ bản của Flutter

2. 1. 2. 1 Các thành phần chính

Ngôn ngữ Dart: Flutter sử dụng Dart, một ngôn ngữ lập trình do Google phát triển. Dart hỗ trợ cả lập trình hướng đối tượng và hàm, mang lại sự linh hoạt và hiệu quả trong việc xây dựng ứng dụng. Điểm nổi bật của Dart là hiệu suất cao, cú pháp rõ ràng, và khả năng biên dịch AOT (Ahead Of Time) và JIT (Just In Time), giúp ứng dụng Flutter chạy mượt mà và nhanh chóng.

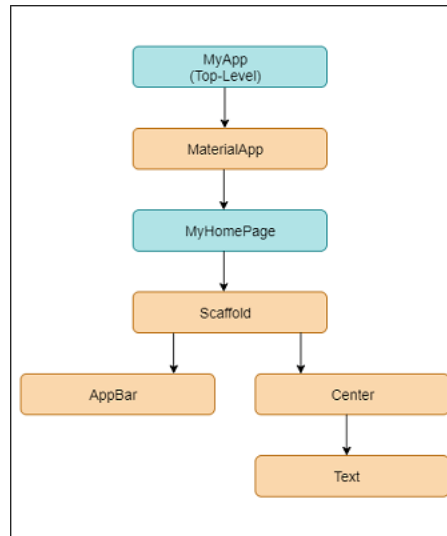
Flutter Engine: Được coi là trái tim của Flutter, viết bằng C++. Engine này làm nhiệm vụ rendering các giao diện người dùng, xử lý các đầu vào (input) và các sự kiện. Nó kết nối trực tiếp với cảm biến và các phần cứng khác của thiết bị, đồng thời là cầu nối giữa code Dart và nền tảng gốc (native) của thiết bị.

Foundation Library: Thư viện cung cấp các API cơ bản và cần thiết cho việc xây dựng ứng dụng. Bao gồm các lớp và hàm để quản lý và tương tác với các thành phần UI.

Design-specific Widgets: Flutter cung cấp một bộ sưu tập lớn các widgets, được thiết kế để tạo ra giao diện người dùng đẹp mắt và tương tác mượt mà. Có widgets cho cả Material Design (Google) và Cupertino (Apple iOS).

2. 1. 2. 2 Các thành phần framework

- Widgets: Các thành phần UI cơ bản.



Hình 2. Widgets

Là các khối cơ bản trong Flutter. Một nút, một văn bản, một hình ảnh, một bố cục,... đều là một widget.

Widgets gồm hai loại chính: Stateless Widgets và Stateful Widgets. Stateless Widgets không thay đổi trạng thái sau khi được tạo, Stateful Widgets thay đổi những gì đang hiển thị bằng cách thay đổi State của chính nó.

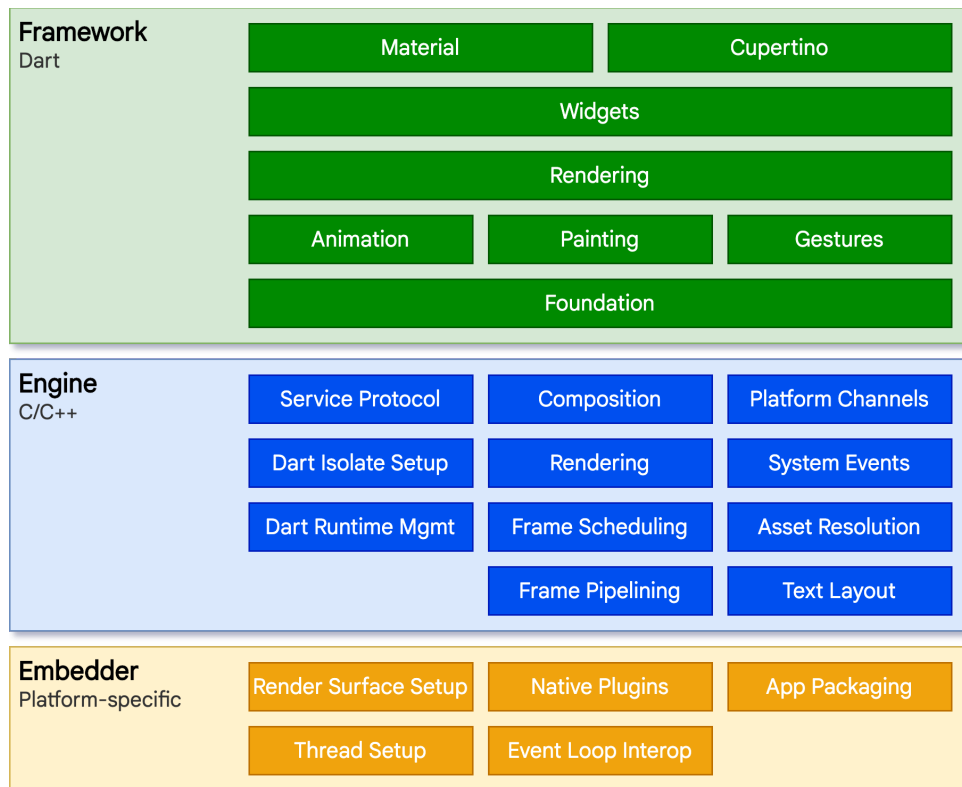
Cấu trúc Widget Tree: Các widgets được tổ chức theo dạng cây, giúp xây dựng giao diện người dùng một cách linh hoạt và dễ quản lý.

- Layers: Xử lý vòng đời của Widget

Layers là một khái niệm quan trọng trong Flutter, layers được phân loại thành nhiều hạng mục dựa trên độ phức tạp và được sắp xếp theo hướng từ trên xuống.

Lớp trên cùng là giao diện người dùng (UI) của ứng dụng, đặc thù cho các nền tảng Android và iOS. Lớp tiếp theo chứa tất cả các widget native của Flutter. Lớp kế tiếp là lớp hiển thị, nơi thực hiện việc render tất cả các thành phần trong ứng dụng Flutter.

Các lớp tầng dưới bao gồm lớp Gesture, Foundation, Engine, và cuối cùng là mã cốt lõi đặc thù cho từng nền tảng.



Hình 3. Layers

- **Rendering:** Render các Widget thành cây UI

Quá trình rendering trong Flutter diễn ra mỗi khi có sự thay đổi trong trạng thái của widgets, đảm bảo rằng giao diện người dùng luôn cập nhật và phản ánh đúng trạng thái hiện tại của ứng dụng.

- **Gesture system:** Xử lý thao tác và cử chỉ người dùng

Flutter cung cấp một hệ thống nhận diện cử chỉ phong phú, cho phép các ứng dụng tương tác một cách tự nhiên và trực quan với người dùng.

Hệ thống này hỗ trợ các cử chỉ như press, drag, pinch để zoom,.... Điều này giúp tạo ra trải nghiệm người dùng tương tác và mượt mà.

Mỗi gesture có thể được xử lý thông qua các widgets cụ thể hoặc thông qua hệ thống cử chỉ toàn cục để tạo ra các phản hồi tương ứng trong ứng dụng.

2. 1. 2. 3 Mô hình hoạt động của Flutter

Flutter hoạt động dựa trên mô hình 'Everything is a widget'. Từ các thành phần giao diện nhỏ nhất (như nút, văn bản) đến cả cấu trúc trang, tất cả đều là

widgets. Các widgets này được tổ chức theo cấu trúc cây, cho phép xây dựng giao diện người dùng một cách linh hoạt và hiệu quả.

Khi một widget thay đổi (do trạng thái thay đổi hoặc tương tác người dùng), Flutter engine sẽ tái tạo giao diện người dùng bằng cách so sánh cây widget mới với cây widget cũ và chỉ cập nhật những phần cần thiết.

2. 1. 3 Ngôn ngữ lập trình Flutter

2. 1. 3. 1 Ngôn ngữ Dart

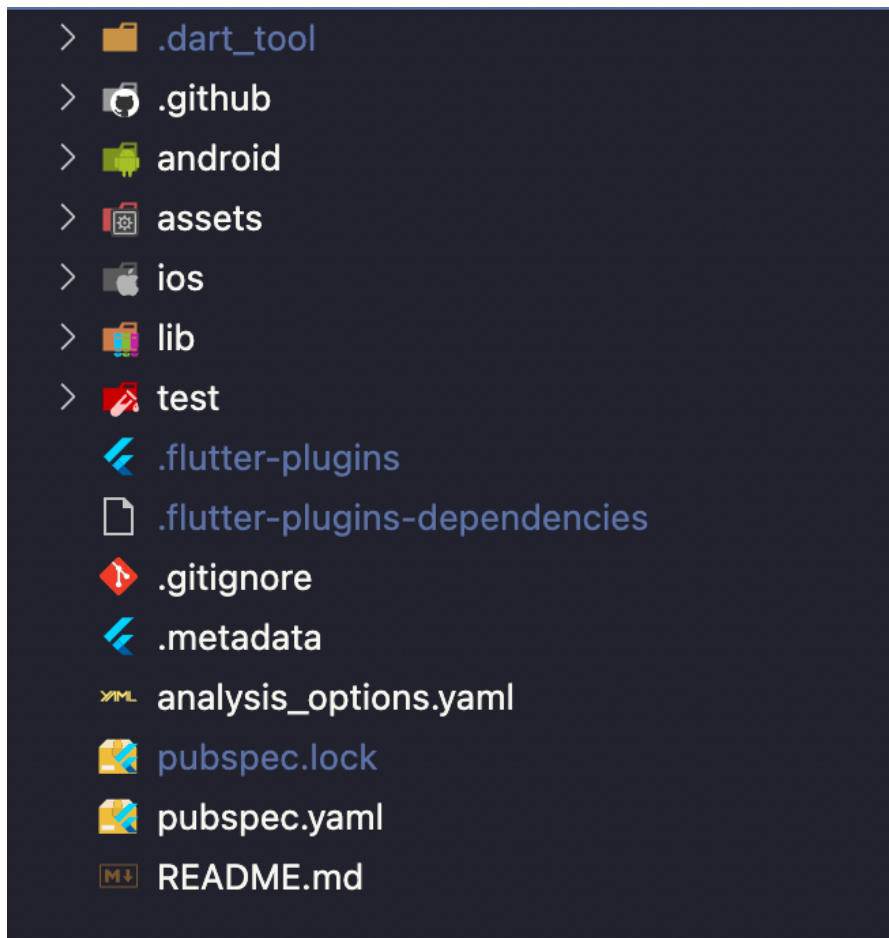


Hình 4. Ngôn ngữ Dart

Dart được phát triển bởi Google, là một ngôn ngữ lập trình mã nguồn mở. Dart thiết kế với cấu trúc hướng đối tượng và cú pháp dựa trên ngôn ngữ C. Dart hỗ trợ đầy đủ các tính chất của lập trình hướng đối tượng, cho phép lập trình viên sáng tạo không giới hạn trong việc phát triển ứng dụng.

Dart cũng nổi tiếng với việc là ngôn ngữ lập trình chính của Flutter, một framework phát triển ứng dụng di động nhanh chóng và linh hoạt của Google. Sự kết hợp này tạo ra một công cụ mạnh mẽ cho việc tạo ra các ứng dụng đa nền tảng với hiệu suất cao và giao diện người dùng mượt mà [2].

2. 1. 3. 2 Cấu trúc thư mục trong dự án Flutter



Hình 5. Cấu trúc thư mục trong dự án Flutter

Thư mục **.dart_tool**: Bên trong gồm 2 thành phần chính

- flutter_build: Lưu trữ các bộ nhớ đệm về bản build của Flutter.
- package_config.json: Nơi chỉ định cụ thể vị trí và phiên bản của những package, dependency hay thư viện mà dự án sử dụng.

Thư mục **android**: Là nơi chứa các cài đặt, tài nguyên cần thiết để chạy ứng dụng Flutter trên hệ điều hành Android.

Thư mục **assets**: Lưu trữ các hình ảnh, phông chữ, ... được sử dụng trong toàn bộ dự án.

Thư mục **ios**: Giống như thư mục android, là nơi chứa các cài đặt, tài nguyên cần thiết để chạy ứng dụng Flutter trên hệ điều hành IOS

Thư mục **lib**:

- Là thư mục quan trọng nhất của dự án, nơi thực hiện xây dựng dự án.

- Mặc định thư mục lib chứa tệp main.dart. Khi chạy dự án tệp main.dart được chạy đầu tiên.

Thư mục **test**: Chứa code kiểm thử phần mềm.

Tệp **pubspec.yaml**: Là nơi chứa cấu hình cụ thể cho ứng dụng như tên, phiên bản, môi trường,...

2. 1. 3. 3 Các tính năng nổi bật của Dart

Hướng đối tượng: Dart là ngôn ngữ lập trình hướng đối tượng, cho phép mô hình hóa các đối tượng thực tế thành các lớp (class) và đối tượng (object), có các thuộc tính và phương thức. Dart hỗ trợ các khái niệm cơ bản trong lập trình OOP như trừu tượng hóa, đóng gói, kế thừa và đa hình.

Bất đồng bộ: Dart hỗ trợ lập trình không đồng bộ thông qua các Future và Stream. Điều này cho phép chạy đồng thời nhiều task một cách độc lập, tăng hiệu suất xử lý.

Các thư viện tích hợp: Dart tích hợp sẵn nhiều thư viện như dart:core, dart:async, dart:math... giúp phát triển ứng dụng nhanh chóng. Người lập trình dễ dàng sử dụng lại code thay vì phải viết lại.

Hỗ trợ đa nền tảng: Nhờ Dart VM và Dart compile to native, Dart có thể chạy trên nhiều nền tảng như Windows, Linux, MacOS, cũng như các mobile platform.

2. 1. 3. 4 Dart hỗ trợ phát triển ứng dụng với Flutter

Ngôn ngữ Dart được thiết kế đặc biệt để hỗ trợ phát triển ứng dụng hiệu quả với Flutter với các đặc điểm sau:

Cú pháp đơn giản, ngắn gọn giúp xây dựng giao diện nhanh chóng dựa trên các Widget.

Dart VM và JIT compiler cho phép biên dịch trực tiếp Dart code thành native code, đem lại hiệu năng cao cho ứng dụng.

Hỗ trợ tính năng Hot Reload, cho phép xây dựng và cập nhật giao diện ngay lập tức mà không cần quá trình biên dịch lâu dài.

Dart Stream API rất hữu ích trong việc quản lý trạng thái và lập trình phản ứng trong Flutter.

2. 1. 4 Quản lý trạng thái ứng dụng

Việc quản lý trạng thái (state) là rất quan trọng trong các ứng dụng Flutter để cập nhật giao diện khi dữ liệu thay đổi. Một số cách quản lý trạng thái:

Sử dụng StatefulWidget và lưu trữ state trong đối tượng state của Widget. Khi state thay đổi, gọi setState() để rebuild UI.

Sử dụng các state management library như BLoC, Redux, MobX để quản lý state ở mức toàn ứng dụng và trigger rebuild UI khi cần thiết.

Lưu trữ state trong các service riêng biệt, và sử dụng Stream để update thay đổi đến các widget. Các widget sẽ lắng nghe sự kiện stream và rebuild khi nhận được dữ liệu mới.

Sử dụng provider package để quản lý và cung cấp các trạng thái dưới dạng provider cho toàn bộ cây widget. Các widget có thể truy xuất và rebuild khi provider thay đổi.

2. 1. 5 Ưu nhược điểm Flutter

2. 1. 5. 1 Ưu điểm Flutter

Flutter cho phép phát triển ứng dụng di động nhanh chóng nhờ sử dụng ngôn ngữ Dart dễ học và trực quan. Flutter có các ưu điểm sau:

Tốc độ phát triển nhanh: Chỉ cần một codebase duy nhất có thể xây dựng ra cả ứng dụng Android và iOS.

Hỗ trợ tính năng hot reload tiện lợi giúp cập nhật giao diện ngay lập tức mà không cần biên dịch.

Cho phép kiểm soát và tùy biến giao diện ở mức độ vô cùng chi tiết. Các hiệu ứng hình ảnh và animation có thể dễ dàng thêm vào.

Áp dụng được cả cho web và các nền tảng desktop như Windows, MacOS.

2. 1. 5. 2 Nhược điểm Flutter

Flutter có một số hạn chế:

Hiệu năng cao, tuy nhiên chưa ổn định bằng các framework đã lâu năm như React Native.

Kích thước ứng dụng nhỏ hơn do Flutter engine có dung lượng lớn.

Một số app yêu cầu native code phức tạp sẽ gặp khó khăn khi dùng Flutter.

2. 2 RESTful API

2. 2. 1 Khái niệm REST và RESTful API

2. 2. 1. 1 Giới thiệu về kiến trúc REST

REST (Representational State Transfer) là một kiến trúc được đề xuất bởi Roy Thomas Fielding năm 2000, áp dụng trong giao tiếp giữa các máy tính quản lý tài nguyên trên internet.

REST sử dụng rộng rãi trong phát triển các ứng dụng dịch vụ Web, giao tiếp qua giao thức HTTP. Các ứng dụng dựa trên kiến trúc REST được gọi là RESTful API.

Kiến trúc REST bao gồm: máy chủ dữ liệu, máy chủ API và máy khách. Từ phía máy khách, các lập trình viên có thể truy cập, thao tác dữ liệu mà không cần quan tâm đến cách hệ thống hoạt động [3].

2. 2. 1. 2 Định nghĩa RESTful API

RESTful API là kiểu API tuân thủ các nguyên tắc và ràng buộc kiến trúc REST.

RESTful API đáp ứng các yêu cầu:

- Giao tiếp qua giao thức HTTP với các phương thức như GET, POST, PUT, DELETE
- Sử dụng URI để xác định resource
- Trả về dữ liệu định dạng JSON hoặc XML
- Tính stateless: mỗi yêu cầu chứa đầy đủ thông tin, không lưu trữ context hoặc session

2. 2. 1. 3 Các ràng buộc trong RESTful API

Client-server: Kiến trúc REST được xây dựng dựa trên mô hình client-server, trong đó server bao gồm các dịch vụ xử lý các yêu cầu từ client. Sự tách biệt này nhằm tăng tính linh hoạt cho client và khả năng mở rộng cho server.

Stateless (phi trạng thái): REST đảm bảo tính stateless, theo đó server và client không lưu trữ trạng thái của nhau. Mỗi yêu cầu và phản hồi chứa đầy đủ thông tin cần thiết, không phụ thuộc lẫn nhau. Điều này nhằm tăng khả năng mở rộng và phát triển hệ thống.

Cacheable (Lưu được vào bộ nhớ cache): Các phản hồi có thể được lưu trong bộ nhớ cache của server để giảm thời gian xử lý. REST yêu cầu các yêu cầu phải mang tính duy nhất để tránh xung đột phản hồi.

Layered system (Hệ thống phân lớp): Hệ thống REST được phân chia thành các tầng, mỗi tầng giao tiếp độc lập. Điều này nhằm mục đích cải thiện khả năng cân bằng tải và lưu trữ đệm dữ liệu.

Uniform interface (chuẩn hoá các interface): REST chuẩn hóa các giao diện để định danh các tài nguyên bằng URI. Đây là ràng buộc quan trọng nhất trong kiến trúc REST. So với các API truyền thống, RESTful API có ưu điểm về hiệu năng, bảo mật và khả năng mở rộng cao nhờ tuân theo mô hình client-server.

2. 2. 2 Các thành phần của RESTful API

2. 2. 2. 1 Resource

Resource là các đối tượng dữ liệu được RESTful API cung cấp và điều khiển truy cập. Resource có thể là một đối tượng (user, sản phẩm) hoặc một tập hợp đối tượng (danh sách user, giỏ hàng). Resource là trọng tâm của RESTful API.

2. 2. 2. 2 URI

URI là một chuỗi ký tự xác định vị trí của một tài nguyên. Ví dụ: /users, /products

Trong RESTful API, mỗi resource sẽ có ít nhất một URI riêng biệt. URI giúp client truy xuất chính xác đến resource mong muốn.

2. 2. 2. 3 Representation

Representation là định dạng dữ liệu của resource khi được truyền tải qua lại giữa client và server. Ví dụ JSON, XML, HTML...

2. 2. 2. 4 Method

Là các phương thức HTTP để thực hiện thao tác với Resource: GET, POST, PUT, DELETE...

2. 2. 3 Các status code trong RESTful API

Status code là mã trạng thái HTTP mà server trả về để cung cấp thông tin về kết quả xử lý request tới resource.

Một số status code thường gặp:

- 200 (OK): Yêu cầu được xử lý thành công.
- 201 (Created): Tạo mới resource thành công.
- 400 (Bad request): Cú pháp request không hợp lệ.
- 401 (Unauthorized): Chưa xác thực hoặc xác thực không hợp lệ.
- 403 (Forbidden): Server từ chối truy cập vào resource.
- 404 (Not found): Không tìm thấy resource.
- 500 (Internal server error): Lỗi server bên trong.

2. 2. 4 Xác thực API

2. 2. 4. 1 Tầm quan trọng của việc xác thực trong RESTful API

Xác thực người dùng là yếu tố then chốt để bảo mật API. Xác thực giúp:

- Ngăn chặn truy cập trái phép vào API và dữ liệu nhạy cảm.
- Phân quyền truy cập các tính năng của API.
- Theo dõi hoạt động của người dùng.

2. 2. 4. 2 Các phương thức xác thực API phổ biến

Một số phương thức xác thực phổ biến:

- Basic Auth: Sử dụng username và password.

- JWT: JSON Web Token.
- OAuth 2.0: Sử dụng token truy cập.

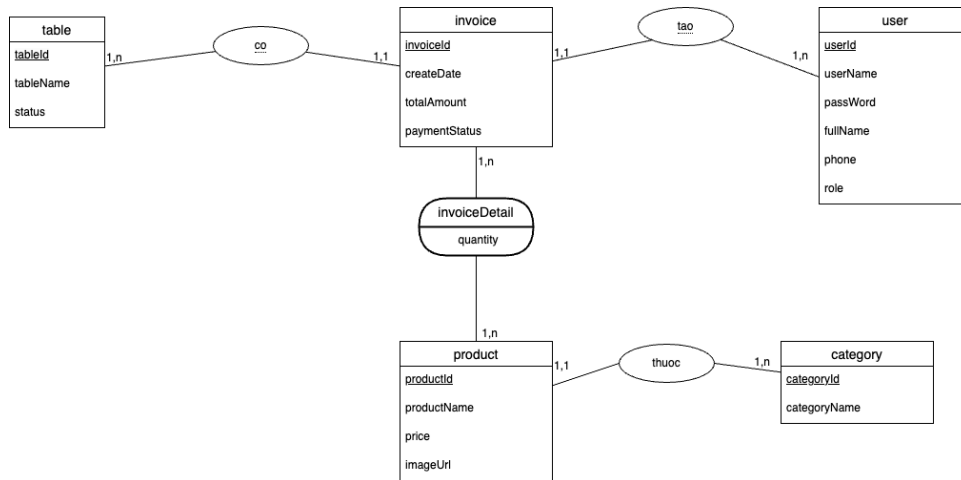
2. 2. 4. 3 Lợi ích của việc xác thực API

- Theo dõi hoạt động của người dùng.
- Thu thập dữ liệu phục vụ phân tích.
- Ngăn chặn spam, abuse.
- Đơn giản hóa quá trình monetization.

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

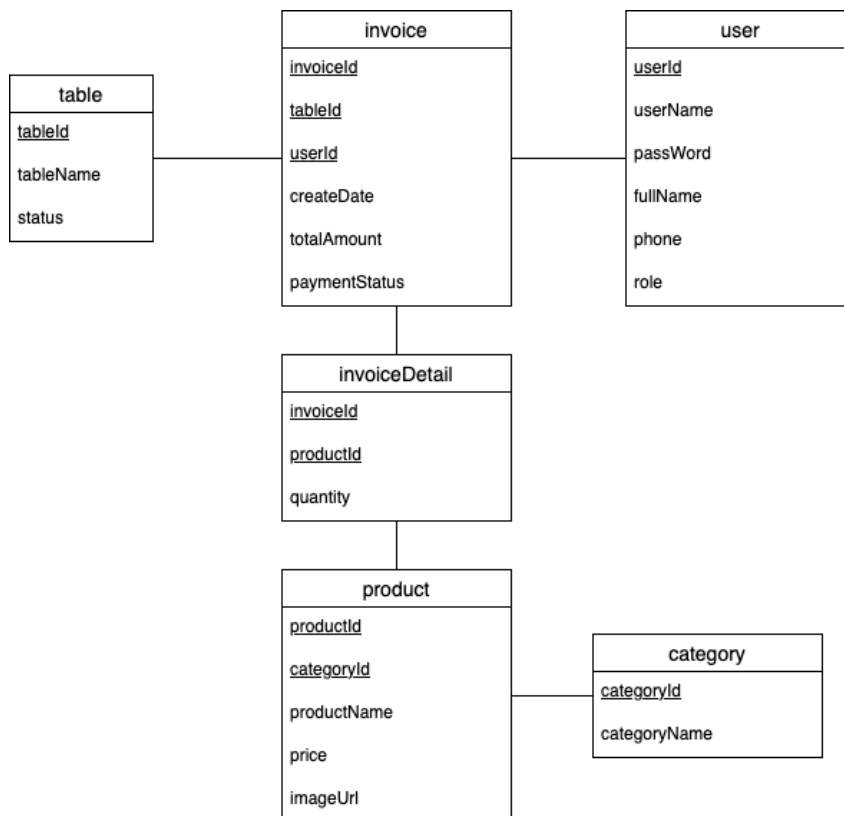
3.1 Đặt tải ứng dụng

3.1.1 Mô hình dữ liệu mức quan hệ



Hình 6. Mô hình dữ liệu mức quan hệ

3.1.2 Mô hình dữ liệu mức vật lý



Hình 7. Mô hình dữ liệu mức vật lý

Bảng 2. Bảng user

| Thuộc tính | Mô tả | Kiểu dữ liệu | Ràng buộc |
|------------|--------------------------|---------------|------------|
| userId | Mã người dùng | Int | Khóa chính |
| userName | Tên đăng nhập người dùng | Varchar (50) | |
| passWord | Mật khẩu người dùng | Varchar (50) | |
| fullName | Họ tên người dùng | Varchar (100) | |
| phone | Số điện thoại người dùng | Varchar (10) | |
| role | Vai trò người dùng | Varchar (10) | |

Bảng 3. Bảng table

| Thuộc tính | Mô tả | Kiểu dữ liệu | Ràng buộc |
|------------|----------------|--------------|------------|
| tableId | Mã bàn | Int | Khóa chính |
| tableName | Tên bàn | Varchar (10) | |
| status | Trạng thái bàn | Varchar (20) | |

Bảng 4. Bảng invoice

| Thuộc tính | Mô tả | Kiểu dữ liệu | Ràng buộc |
|---------------|-----------------------|--------------|------------|
| invoiceId | Mã hóa đơn | Int | Khóa chính |
| tableId | Tên bàn | int | Khóa ngoại |
| userId | Trạng thái bàn | int | Khóa ngoại |
| createDate | Ngày tạo hóa đơn | date | |
| totalAmount | Tổng tiền hóa đơn | float | |
| paymentStatus | Trạng thái thanh toán | Varchar (20) | |

Bảng 5. Bảng category

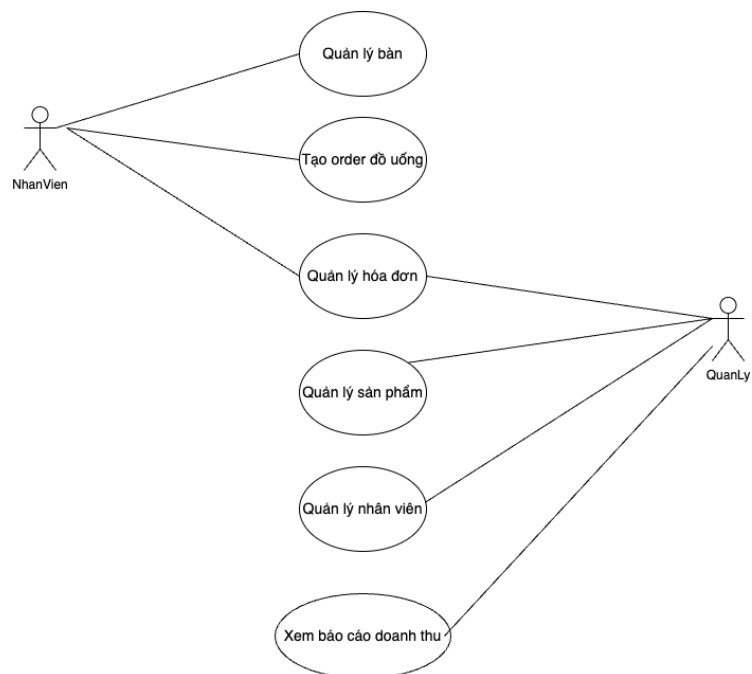
| Thuộc tính | Mô tả | Kiểu dữ liệu | Ràng buộc |
|--------------|----------|--------------|------------|
| categoryId | Mã loại | Int | Khóa chính |
| categoryName | Tên loại | Varchar (20) | |

Bảng 6. Bảng product

| Thuộc tính | Mô tả | Kiểu dữ liệu | Ràng buộc |
|-------------|-------------------|---------------|------------|
| productId | Mã sản phẩm | Int | Khóa chính |
| categoryId | Mã loại | int | Khóa ngoại |
| productName | Tên sản phẩm | Varchar (50) | |
| price | Giá sản phẩm | float | |
| imageUrl | Hình ảnh sản phẩm | Varchar (255) | |

3. 1. 3 Sơ đồ Use-case

3. 1. 3. 1 Sơ đồ



Hình 8. Sơ đồ Use-case

3. 1. 3. 2 Danh sách Actor

Bảng 7. Bảng danh sách Actor

| STT | Tên Actor | Ý nghĩa |
|-----|-----------|------------------------------------|
| 1 | NhanVien | Nhân viên tạo order cho khách hàng |
| 2 | QuanLy | Quản lý ứng dụng |

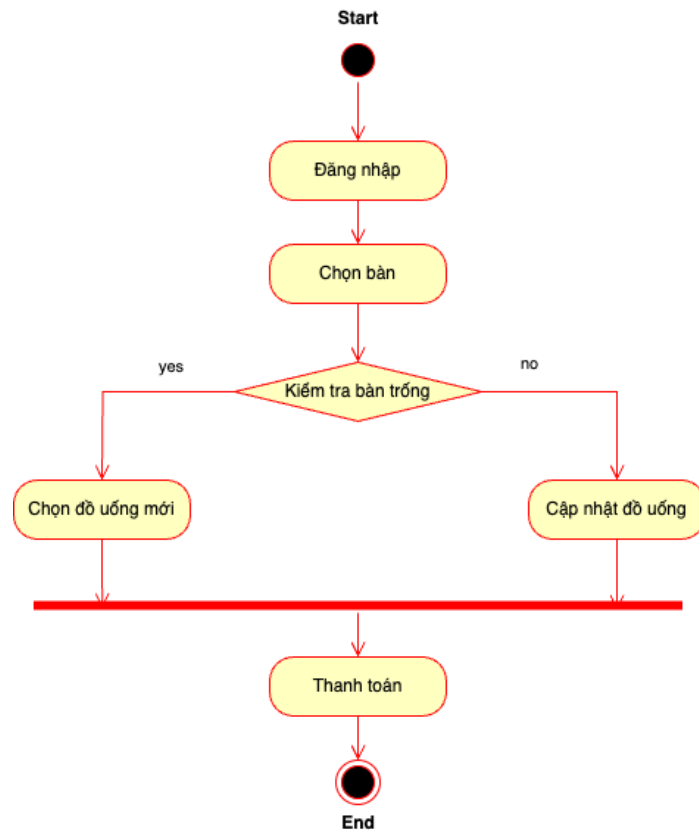
3. 1. 3. 3 Danh sách Use-case

Bảng 8. Bảng danh sách Use-case

| STT | Tên Use-case | Ý nghĩa |
|-----|-----------------------|---|
| 1 | Quản lý bàn | Xem bàn trống hay đã có người |
| 2 | Tạo order đồ uống | Tiếp nhận order, gọi đồ uống cho khách, có thể gọi nhiều món cùng lúc |
| 3 | Quản lý hóa đơn | Tạo hóa đơn khi khách hàng thanh toán |
| 4 | Quản lý sản phẩm | Quản lý danh mục, thông tin đồ uống |
| 5 | Quản lý nhân viên | Quản lý thông tin nhân viên |
| 6 | Xem báo cáo doanh thu | Thống kê doanh thu |

3. 1. 4 Sơ đồ hoạt động

Sơ đồ hoạt động order đồ uống



Hình 9. Sơ đồ order đồ uống

Hoạt động diễn ra khi khách hàng đến và có nhu cầu gọi đồ uống.

Đăng nhập: Người dùng cần đăng nhập vào hệ thống trước khi thực hiện bất kỳ hoạt động nào.

Chọn bàn: Người dùng chọn bàn theo vị trí bàn khách hàng đang ngồi

Kiểm tra bàn trống:

- Nếu bàn đang trống ("yes"), người dùng tiến hành chọn đồ uống mới để thêm vào đơn hàng.

- Nếu bàn không trống ("no"), nghĩa là có một đơn hàng đang được xử lý tại bàn đó, và người dùng sẽ cập nhật đồ uống cho đơn hàng hiện tại.

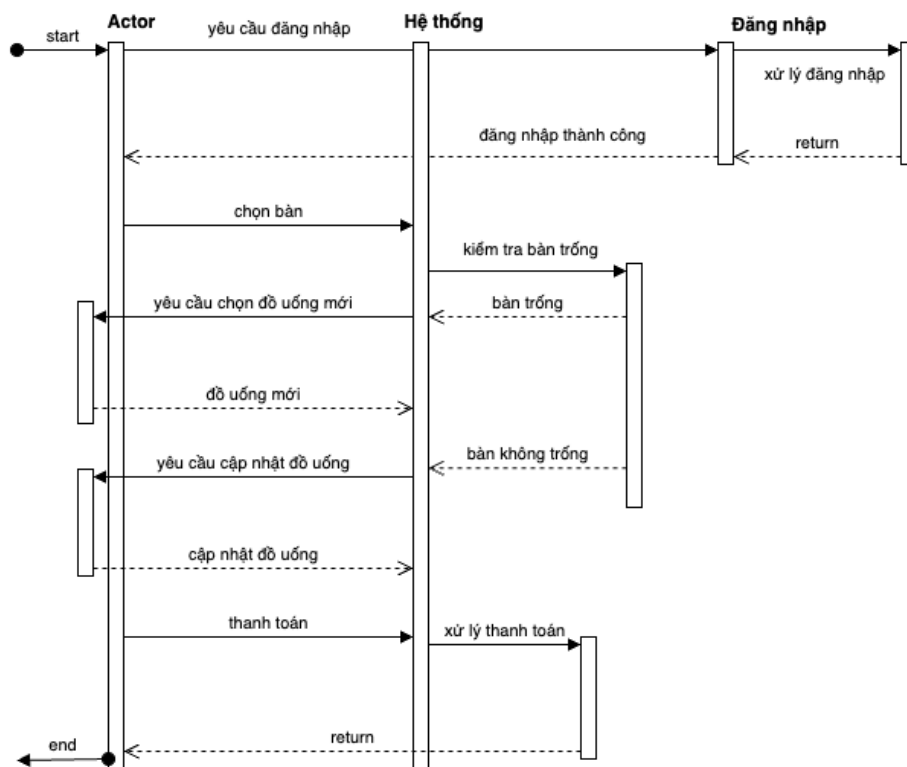
Chọn đồ uống mới: Người dùng thêm một hoặc nhiều đồ uống mới vào đơn hàng.

Cập nhật đồ uống: Người dùng cập nhật đơn hàng bằng cách thay đổi số lượng hoặc thêm đồ uống mới trong đơn hàng hiện tại.

Thanh toán: Sau khi hoàn tất việc chọn đồ uống mới hoặc cập nhật đồ uống, người dùng tiến hành thanh toán đơn hàng đơn hàng cho khách hàng.

Quy trình kết thúc sau khi việc thanh toán được hoàn tất.

3. 1. 5 Sơ đồ tuần tự



Hình 10. Sơ đồ tuần tự order đồ uống

Người dùng (Actor): bắt đầu quy trình bằng cách gửi yêu cầu đăng nhập đến hệ thống.

Hệ thống:

- Xử lý đăng nhập: Hệ thống nhận yêu cầu và xử lý thông tin đăng nhập.
- Đăng nhập thành công: Hệ thống trả về thông điệp đăng nhập thành công đến người dùng.

Chọn bàn:

- Người dùng sau đó chọn một bàn.
- Hệ thống kiểm tra xem bàn đó có đang trống hay không.

Đặt đồ uống:

- Nếu bàn trống, hệ thống yêu cầu người dùng chọn đồ uống mới.
- Người dùng gửi yêu cầu đồ uống mới đến hệ thống.
- Nếu bàn không trống, hệ thống yêu cầu cập nhật đồ uống.
- Người dùng gửi yêu cầu cập nhật đồ uống đến hệ thống.

Thanh toán:

- Người dùng yêu cầu thanh toán cho đơn hàng.
- Hệ thống xử lý yêu cầu thanh toán.

Kết thúc:

- Hệ thống trả về thông điệp hoàn tất (return), và quy trình kết thúc tại điểm này (end).

3. 2 Kiến trúc hệ thống

3. 2. 1 Client (Flutter)

Sử dụng framework Flutter để xây dựng giao diện cho ứng dụng bao gồm các màn hình: đăng nhập, danh sách bàn, menu, giỏ hàng, quản lý hóa đơn.

Sử dụng các package như **http** để thực hiện các yêu cầu HTTP đến server và xử lý dữ liệu trả về.

Sử dụng Provider, Bloc để quản lý trạng thái ứng dụng.

Authentication: Sử dụng Firebase Authentication để xác thực người dùng.

Database: Sử dụng Cloud Firestore để lưu trữ và quản lý dữ liệu dưới dạng tài liệu và collection, bao gồm thông tin về bàn, sản phẩm, hóa đơn, và thông tin người dùng.

Storage: Dùng Firebase Storage để lưu trữ hình ảnh và các tệp tin.

Firebase Cloud Messaging: Để gửi thông báo đẩy đến người dùng.

3. 2. 2 Firebase Backend

No Server Management: Không cần quản lý server riêng, Firebase cung cấp một nền tảng đầy đủ các dịch vụ backend.

Firebase SDK: Tích hợp trực tiếp với ứng dụng Flutter thông qua các Firebase SDK.

3. 3 Thiết kế RESTful API

Trong Firebase, thay vì sử dụng các method GET, POST, PUT, DELETE truyền thống như trong RESTful API, chúng ta sẽ làm việc trực tiếp với Cloud Firestore database qua các SDK của Firebase. Firestore là một cơ sở dữ liệu NoSQL dựa trên tài liệu (documents) và bộ sưu tập (collections). Mỗi tài liệu có thể chứa các trường dữ liệu cùng với một ID duy nhất [4].

Dưới đây là cách thiết kế cơ bản cho các thực thể hóa đơn(invoice):

3. 3. 1 Đọc hóa đơn



```

FirebaseFirestore.instance
  .collection('invoices')
  .get()
  .then((QuerySnapshot querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      // Xử lý dữ liệu hóa đơn
    });
  });

```

Hình 11. Đọc hóa đơn

3. 3. 2 Đọc hóa đơn theo Id

```

FirebaseFirestore.instance
  .collection('invoices')
  .doc('invoiceId')
  .get()
  .then((DocumentSnapshot documentSnapshot) {
    if (documentSnapshot.exists) {
      // Xử lý dữ liệu hóa đơn
    } else {
      // Xử lý trường hợp không tìm thấy hóa đơn
    }
  });

```

Hình 12. Đọc hóa đơn theo Id

3. 3. 3 Tạo hóa đơn mới

```

FirebaseFirestore.instance
  .collection('invoices')
  .add({
    'tableId': tableId,
    'userId': userId,
    'createDate': createDate,
    'totalAmount': totalAmount,
    'paymentStatus': paymentStatus,
  })
  .then((DocumentReference doc) {
    // Xử lý sau khi tạo hóa đơn thành công
  })
  .catchError((e) {
    // Xử lý lỗi
  });

```

Hình 13. Tạo hóa đơn mới

3. 3. 4 Cập nhật hóa đơn

```

FirebaseFirestore.instance
  .collection('invoices')
  .doc('invoiceId')
  .update({
    'tableId': newTableId,
    'totalAmount': newTotalAmount,
    'paymentStatus': newPaymentStatus,
  })
  .then((_) {
    // Xử lý sau khi cập nhật thành công
  })
  .catchError((e) {
    // Xử lý lỗi
  });

```

Hình 14. Cập nhật hóa đơn

3.3.5 Xóa hóa đơn



```

FirebaseFirestore.instance
  .collection('invoices')
  .doc('invoiceId')
  .delete()
  .then((_) {
    // Xử lý sau khi xóa thành công
  })
  .catchError((e) {
    // Xử lý lỗi
  });

```

Hình 15. Xóa hóa đơn

3.4 Tích hợp Flutter với RESTful API

3.4.1 Thêm dependency

Đầu tiên, thêm package 'http' vào file 'pubspec.yaml' của dự án Flutter:

dependencies:

flutter:

 sdk: flutter

 http: ^1.1.0

Sau đó, chạy `flutter pub get` để cài đặt package.

3.4.2 Import package

Trong file Dart có dùng package http thực hiện import package http

```
import 'package:http/http.dart' as http;
```

3.4.3 Gửi request GET

Để gửi một request GET và nhận dữ liệu:

```
Future<void> fetchData() async {
  final response = await http.get(Uri.parse('https://your-api-url.com/data'));

  if (response.statusCode == 200) {
    // Nếu server trả về mã trạng thái "OK",
    // parse và sử dụng dữ liệu
    final data = jsonDecode(response.body);
    // Cập nhật UI hoặc state của bạn tại đây
  } else {
    // Nếu server không trả về mã trạng thái "OK",
    // throw an exception.
    throw Exception('Failed to load data');
  }
}
```

Hình 16. Gửi request GET

3. 4. 4 Gửi Request POST

Để gửi một request POST

```
Future<void> createData(Map<String, dynamic> data) async {
  final response = await http.post(
    Uri.parse('https://your-api-url.com/data'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(data),
  );

  if (response.statusCode == 201) {
    // Nếu dữ liệu được tạo thành công trên server,
    // parse và sử dụng dữ liệu
    final data = jsonDecode(response.body);
    // Cập nhật UI hoặc state của bạn tại đây
  } else {
    // Nếu server không tạo được dữ liệu,
    // throw an exception.
    throw Exception('Failed to create data');
  }
}
```

Hình 17. Gửi request POST

3. 4. 5 Hiển Thị Dữ Liệu

Hiển thị dữ liệu trên giao diện bằng cách sử dụng StatefulWidget và cập nhật state khi dữ liệu được tải xong:

```

// Giả sử bạn đã có một biến state lưu trữ dữ liệu
List<MyData> myDataList = [];

// Sau khi nhận dữ liệu, cập nhật state
setState(() {
  myDataList = fetchedData;
});
```

Hình 18. Hiển thị dữ liệu

Sau đó, sử dụng `ListView.builder` hoặc widget tương tự để hiển thị dữ liệu:

```

ListView.builder(
  itemCount: myDataList.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(myDataList[index].title),
      // Thêm các widget khác để hiển thị dữ liệu
    );
  },
)
```

Hình 19. Sử dụng `ListView.builder` để hiển thị dữ liệu

3. 4. 6 Bắt Lỗi và Xử Lý Mã Trạng Thái

```



try {
  final response = await http.get(Uri.parse('https://your-api-url.com/data'));
  // Xử lý response ở đây
} catch (e) {
  // Xử lý lỗi nếu không thể kết nối đến API
  print(e.toString());
}
```

Hình 20. Xử lý mã trạng thái

CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU

4. 1 Giao diện

4. 1. 1 Giao diện đăng nhập



**Hello
Welcome Back**

USERNAME

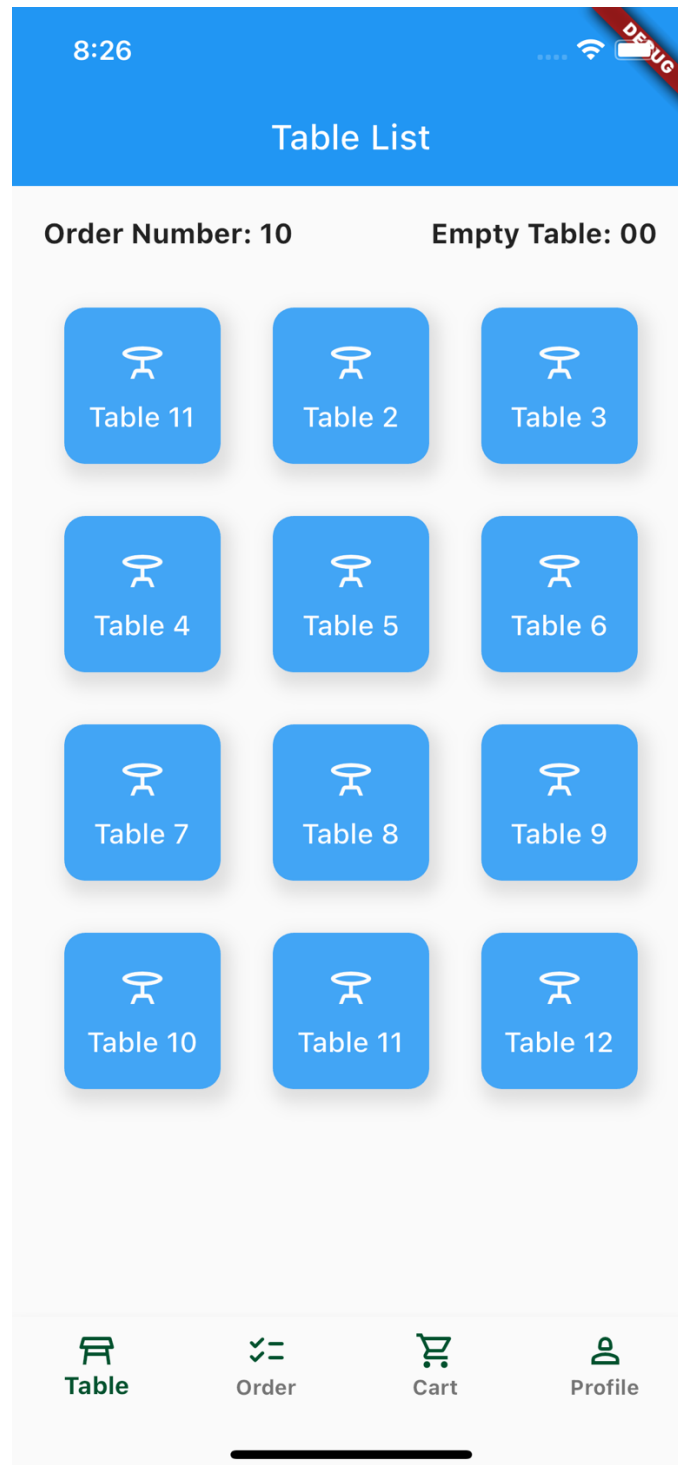
PASSWORD [SHOW](#)

SIGN IN

[NEW USER? SIGN UP](#) [FORGOT PASSWORD?](#)

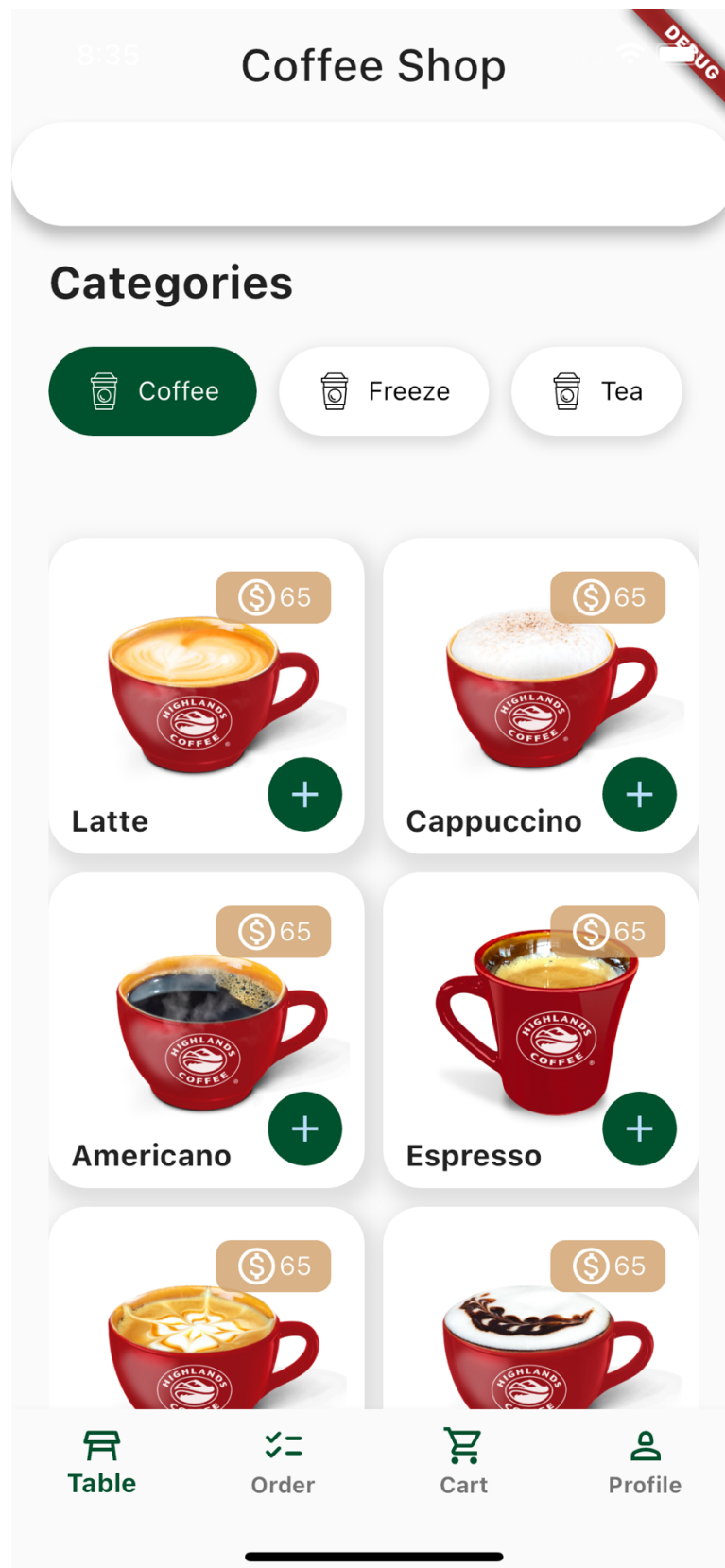
Hình 21. Giao diện đăng nhập

4. 1. 2 Giao diện danh sách bàn



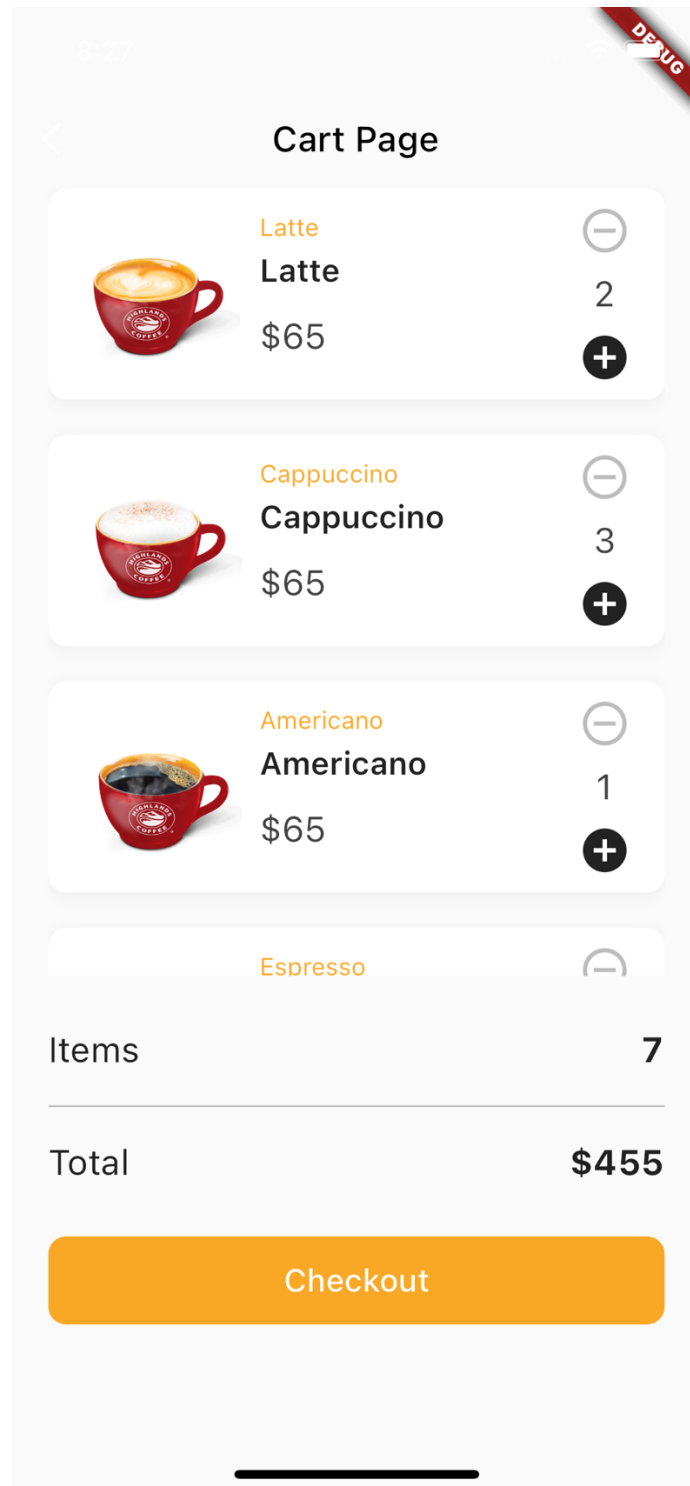
Hình 22. Giao diện danh sách bàn

4. 1. 3 Giao diện danh sách sản phẩm



Hình 23. Giao diện danh sách sản phẩm

4. 1. 4 Giao diện chi tiết order



Hình 24. Giao diện chi tiết order

4. 2 Chức năng

4. 2. 1 Quản lý bàn

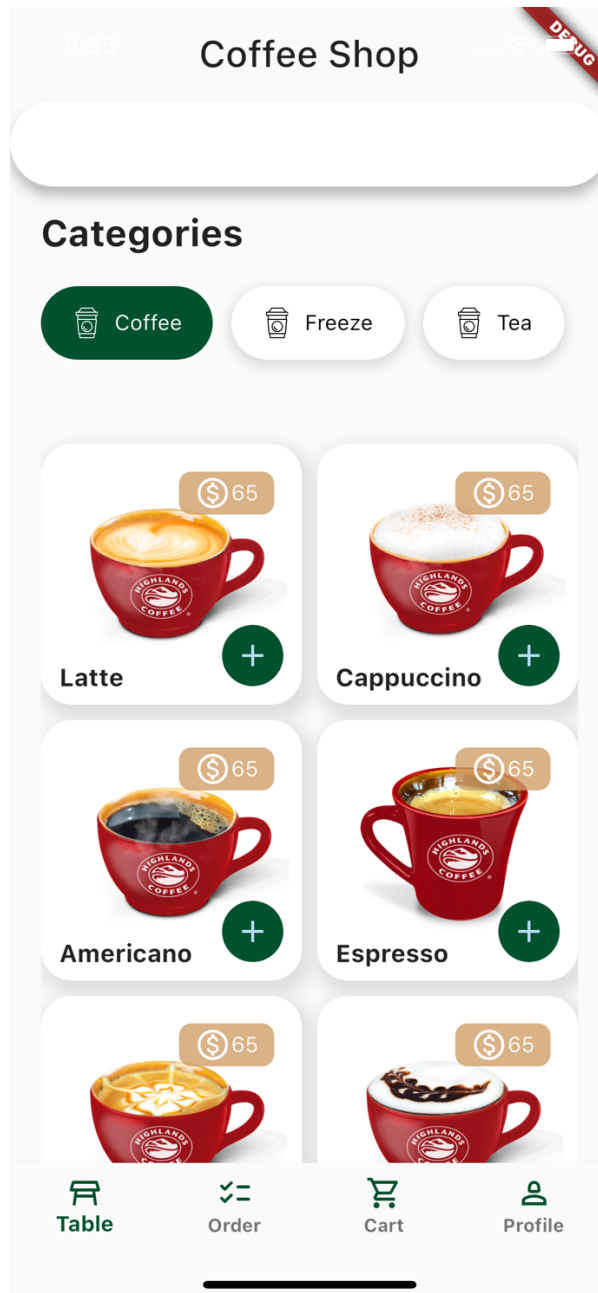


Hình 25. Hiển thị danh sách bàn

Hiển thị danh sách bàn và trạng thái của bàn, hiển thị thông tin số lượng đơn hàng và số lượng bàn đang trống.

Nhấn vào các nút bàn để chuyển qua màn hình chọn đồ uống.

4. 2. 2 Quản lý danh sách sản phẩm

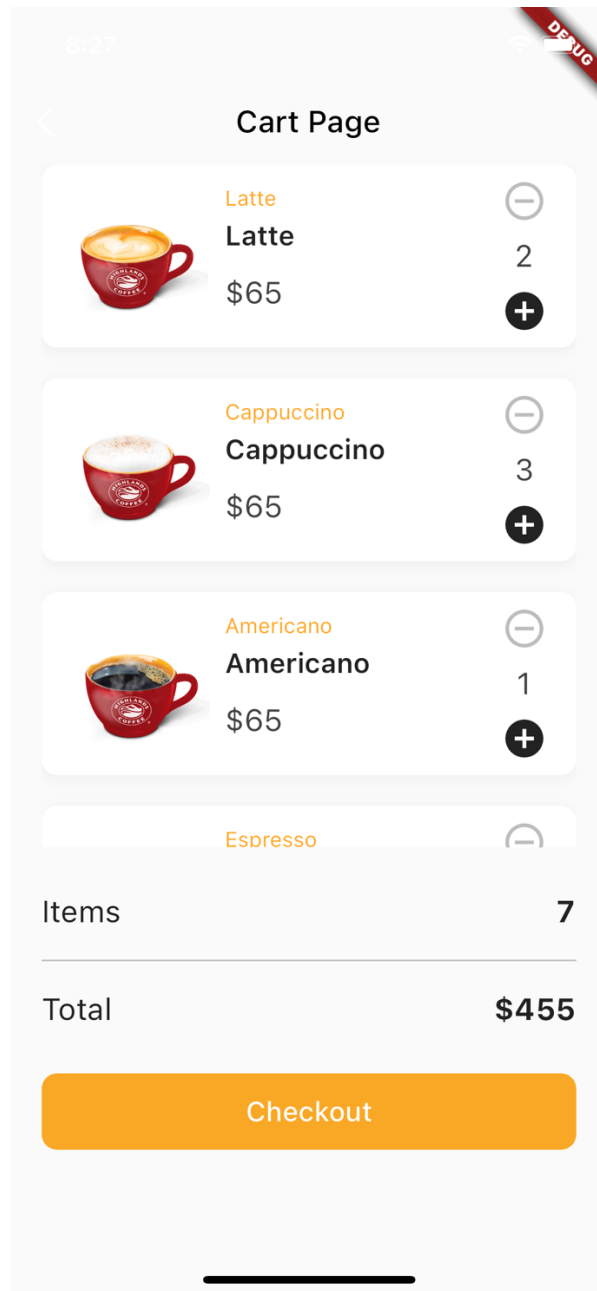


Hình 26. Hiển thị danh sách sản phẩm

Hiển thị danh sách sản phẩm theo thể loại: tên sản phẩm, hình ảnh, giá.

Nhấn nút dấu + để thêm sản phẩm vào đơn hàng.

4. 2. 3 Chi tiết order



Hình 27. Chi tiết order

Thông tin từng sản phẩm của đơn hàng: hình ảnh, tên, giá, số lượng sản phẩm.

Items: Tổng số lượng các sản phẩm trong của đơn hàng.

Total: Tổng tiền của đơn hàng.

Nhấn nút -, + để tăng giảm số lượng sản phẩm.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5. 1 Kết luận

5. 1. 1 Kết quả đạt được

Thực Hiện Các Chức Năng Cơ Bản: quản lý bàn, gọi đồ uống, quản lý đơn hàng.

Hạn Chế ở chức năng Admin: chức năng quản trị viên như quản lý nhân viên, báo cáo doanh thu chưa hoàn thiện.

5. 1. 2 Đóng góp mới

Tích hợp với Flutter, quản lý dữ liệu, và xử lý xác thực người dùng.

5. 1. 3 Đề xuất mới

Mở rộng thực đơn: Đề xuất việc thêm nhiều loại sản phẩm mới, như các loại bánh, để đa dạng hóa lựa chọn cho khách hàng.

Triển khai dịch vụ giao hàng: Phát triển chức năng giao hàng trong ứng dụng, mở rộng phạm vi phục vụ khách hàng.

5. 2 Hướng phát triển:

Hoàn thiện chức năng quản trị: tập trung vào việc cải thiện và hoàn thiện các chức năng quản trị, như quản lý nhân viên, xem báo cáo doanh thu.

Triển khai ứng dụng trên đa nền tảng: mở rộng ứng dụng để hỗ trợ trên nền tảng web hoặc desktop, giúp quản lý dễ dàng hơn từ nhiều thiết bị khác nhau.

Quản lý size sản phẩm: cho phép người dùng lựa chọn kích cỡ hoặc phân loại khác nhau cho các sản phẩm, như kích cỡ cốc cà phê hay loại bánh.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] M. L. Napoli, Beginning Flutter: A Hands On Guide to App Development, 2019.
- [2] N. Metzler, Dart Programming for Beginners: An Introduction to Learn Dart Programming with Tutorials and Hands-On Examples, 2022.
- [3] S. Alessandria, Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart, 2021.
- [4] M. A. S. R. Leonard Richardson, RESTful Web APIs: Services for a Changing World, 2013.
- [5] R. Agarwal, Firebase for Flutter Developers: Authentication, Database and Storage Mastery, 2023.