

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | NHÓM 6 | | |  | | | |
|  |  |  | | | |  | | |
|  |  | FOOD APP | | | |  | | |
|  |  | | | | | |  | |
|  | |  |  | | --- | --- | | Thành Viên | Mã Sinh Viên | | 1. Lê Đức Ngọc | * 2121051437 | | 1. Đỗ Trung Tấn | * 2121050873 | | 1. Trịnh Đức Anh | * 2121051062 |   1. Lời nói đầu  Cùng nhìn lại về quá khứ một chút, vào đại dịch COVID-19 các lệnh hạn chế được ban hành, việc đi lại của mọi người cũng dần bị hạn chế cùng với đó là sự quan ngại về dịch bệnh nên từ đó các dịch vụ giao hang tại nhà ngày càng phát triển. Cho đến ngày nay các app giao hàng đang không ngừng phát triển như Grap, Bee, Shoppe,…  Từ sự phát triển về các dịch vụ giao đồ nhóm 6 lấy ý tưởng từ đó để phát triển một ứng dụng di động nhằm cung cấp dịch vụ giao đồ ăn. Phần mềm giúp người dùng tìm kiếm món ăn, đặt món ăn, thanh toán, them món yêu thích,…  2. Mô tả chức năng của chương trình  Phát triển một ứng dụng chạy trên thiết bị di động Android thực hiện các chức năng tìm kiếm, đặt món, thanh toán,…   * Hiển thị thông báo * Thông tin người dùng   3. Nội dung chức năng   * Đăng nhập, đăng ký * Khi khởi động ứng dụng sẽ cần phải sử dụng tài khoản và mật khẩu để đăng nhập, nếu chưa có tài khoản có thể sử dụng chức năng đăng ký * Giao diện màn hình chính, tìm kiếm * Nhập từ khoá để tìm kiếm sẽ hiện ra list các món ăn có liên quan đến từ khoá bạn đưa ra * Chọn vào món ăn muốn đặt để hiển ra chi tiết về món ăn cũng như về nhà hàng * Lựa chọn đặt món để hiển thị thông tin đơn hàng như về số lượng, giá, phí ship * Danh sách món ăn đã đặt * Hiển thị tất cả các món ăn mà người dùng đã đặt * Thông tin về thời gian địa điểm đặt hàng, giá mà người dùng cần thanh toán * Danh sách món ăn yêu thích * Danh sách thông báo * Thông tin người dùng | | | | | |  | |
|  | Đăng Nhập Đăng Ký I. Giới Thiệu  A. Widget Login và State \_LoginState   * Login: StatefulWidget chứa trạng thái thay đổi được. * LoginState: Quản lý trạng thái của Login.   II. Quá Trình Tải và Đăng Nhập  A. Quá Trình Tải (LoadLogin method)   * Sử dụng LoadStatus để theo dõi trạng thái tải. * LoadLogin giả định một thời gian chờ và chuyển trạng thái tải thành thành công.   B. Xử Lý Đăng Nhập (LoginHandle method)   * Lấy thông tin từ các trường đầu vào. * Gửi yêu cầu POST đến API đăng nhập với thông tin người dùng. * Xử lý phản hồi: * Nếu thành công, lưu trạng thái và chuyển đến màn hình tìm kiếm thức ăn. * Nếu thất bại, hiển thị thông báo lỗi và cập nhật trạng thái.   III. Giao Diện Người Dùng  A. Màn Hình Đăng Nhập   * Hiển thị màn hình tải (InitializationPage) nếu đang tải. * Ngược lại, hiển thị biểu mẫu đăng nhập với các trường và nút tương ứng. * Hiển thị thông báo lỗi nếu có.   B. Giao Diện Đăng Nhập Trường nhập email/số điện thoại.   * Trường nhập mật khẩu. * Nút đăng nhập với chỉ báo tải hoặc văn bản "Đăng nhập". * Liên kết chuyển đến màn hình Đăng ký. * Hiển thị thông báo lỗi nếu có. | | |  | | | | |
| Home Tìm Kiếm I. Giới Thiệu   * Widget SearchFood và State \_SearchFoodState SearchFood: StatefulWidget chứa trạng thái thay đổi được. * \_SearchFoodState: Quản lý trạng thái của SearchFood.   II. Tải Dữ Liệu và Hiển Thị Thông Tin Người Dùng  A. Phương thức fetchData   * Lấy ID người dùng từ SharedPreferences. * Gửi yêu cầu GET đến API để lấy thông tin người dùng. * Hiển thị thông tin người dùng hoặc xử lý lỗi.   III. Giao Diện Người Dùng   1. Màn Hình Tìm kiếm Thực phẩm  * Hiển thị hình nền với hiệu ứng mờ. * Hiển thị chỉ báo tải nếu đang tải, ngược lại hiển thị giao diện tìm kiếm. * Hiển thị thông tin người dùng, tiêu đề chào mừng, và mô tả tìm kiếm.  1. Phần Tìm kiếm  * Trường nhập từ khóa tìm kiếm. * Nút tìm kiếm với sự kiện chuyển hướng đến màn hình kết quả tìm kiếm.  1. Thông Tin Người Dùng  * Hiển thị hình đại diện và tên người dùng. * Hiển thị mô tả chào mừng. | | | |  | | | | |
|  |  | | | | | |  | |
|  |  | | | | | |  | |

# Thực Đơn

I. Giới Thiệu

1. A screenshot of a food item

   Description automatically generatedWidget RestaurantMenuScreen và State RestaurantMenuScreenState

* RestaurantMenuScreen: StatefulWidget chứa trạng thái thay đổi được.
* \_RestaurantMenuScreenState: Quản lý trạng thái của RestaurantMenuScreen.

II. Tải Dữ Liệu và Hiển Thị Thực Đơn

1. Phương thức fetchData

* Xây dựng yêu cầu API để lấy danh sách món ăn.
* Gửi yêu cầu GET đến API và xử lý phản hồi:
* Nếu thành công, cập nhật trạng thái và danh sách món ăn.
* Nếu thất bại, hiển thị thông báo lỗi.

III. Giao Diện Người Dùng

1. Màn Hình Thực Đơn Nhà Hàng

* Thanh hiển thị kết quả tìm kiếm
* Hình ảnh món ăn

1. Danh Sách Món Ăn

* Hiển thị danh sách món ăn sử dụng ListView.builder.
* Mỗi món ăn được hiển thị bằng ItemRestaurantMenuWidget với thông tin như tên, giá, nguyên liệu, mô tả, hình ảnh thumbnail, và số lượng đơn đặt hàng.

# Thông Tin Món Ăn

1. A screenshot of a phone

   Description automatically generatedImports:

* Mã bắt đầu với việc nhập các gói và phụ thuộc cần thiết, bao gồm dart:convert để mã hóa/ giải mã JSON, flutter\_dotenv để quản lý biến môi trường, flutter/material.dart cho các thành phần giao diện người dùng Flutter, và các phụ thuộc tùy chỉnh khác liên quan đến dự án.

1. Lớp DishDetailsScreen:

* Lớp này đại diện cho màn hình hiển thị chi tiết về một món ăn cụ thể.
* Nó mở rộng từ StatefulWidget và nhận tham số idProduct.

1. Lớp \_DishDetailsScreenState:

* Lớp này đại diện cho trạng thái của widget DishDetailsScreen.
* Nó có một đối tượng foodItem kiểu Food và một biến foodLoadStatus kiểu LoadStatus để quản lý trạng thái tải dữ liệu.

1. Phương thức initState:

* Phương thức initState được gọi khi đối tượng trạng thái được chèn vào cây.
* Nó khởi tạo biến idFood bằng idProduct được cung cấp và gọi phương thức getDishDetail để lấy chi tiết về món ăn.

1. Phương thức getDishDetail:

* Phương thức getDishDetail thực hiện một yêu cầu HTTP GET đến một đầu điểm API cụ thể để lấy chi tiết về một món ăn dựa trên ID (idFood).
* Nếu yêu cầu thành công (mã trạng thái 200), dữ liệu phản hồi được giải mã từ JSON và foodItem được cập nhật với dữ liệu lấy được.
* Nếu có lỗi, foodLoadStatus được đặt để thông báo về một lỗi.

1. Phương thức Build:

* Phương thức build chịu trách nhiệm xây dựng giao diện người dùng của màn hình.
* Giao diện người dùng bao gồm nhiều widget khác nhau, bao gồm Container, Padding, Text, Image, InkWell, ListView, và nhiều hơn nữa.
* Giao diện người dùng được hiển thị theo điều kiện dựa trên foodLoadStatus, hiển thị chỉ báo tải, thông báo lỗi hoặc chi tiết về món ăn.
* Chi tiết món ăn bao gồm hình ảnh, tên, tổng số đơn đặt hàng, giá, giờ mở cửa, một nút để đặt hàng và thông tin bổ sung.

1. GestureDetector và Điều hướng:

* Một GestureDetector được sử dụng để phát hiện nhấp chuột vào nút "ĐẶT HÀNG NGAY", và nó chuyển hướng đến OrderProcessingScreen khi được nhấp. idProduct được truyền đến màn hình mới.

1. Styling:

* Mã bao gồm kiểu dáng cho nhiều thành phần giao diện người dùng, chẳng hạn như phông chữ văn bản, màu sắc và padding.

1. Địa phương hóa:

* Mã sử dụng lớp NumberFormat để định dạng giá trong đồng Việt Nam.

1. Xử lý Lỗi:

* Lỗi trong quá trình yêu cầu HTTP hoặc các quy trình khác được bắt, và foodLoadStatus được cập nhật để thông báo về một lỗi. Chi tiết lỗi cũng được in ra console.

1. Ghi chú Bổ sung:

* Mã bao gồm các phần để hiển thị mô tả của món ăn và cung cấp ghi chú hoặc lời khuyên hữu ích.
* Giao diện người dùng là có thể đáp ứng, điều chỉnh bố cục của mình dựa trên trạng thái tải.

# Đặt Món Ăn

* A screenshot of a menu

  Description automatically generatedImport: Các gói và tệp tin cần thiết được nhập vào. Đây bao gồm các gói cho Flutter, yêu cầu HTTP, shared preferences và các lớp mô hình (Food và Order).
* Định nghĩa Widget: Lớp OrderProcessingScreen được định nghĩa là một StatefulWidget, chấp nhận một tham số tùy chọn là idProduct.
* Lớp State: Lớp \_OrderProcessingScreenState mở rộng từ State và chứa logic để lấy thông tin thực phẩm, hiển thị chúng và xử lý đặt hàng.
* Khởi tạo Trạng thái: Phương thức initState được ghi đè để khởi tạo trạng thái khi widget được tạo. Nó đặt idFood từ tham số của widget và gọi phương thức fetchData.
* Phương thức fetchData: Phương thức này thực hiện một yêu cầu HTTP GET để lấy chi tiết thực phẩm dựa trên idFood. Dữ liệu nhận được được sử dụng để cập nhật trạng thái.
* Phương thức pushData: Phương thức này chịu trách nhiệm đẩy dữ liệu đơn hàng lên máy chủ. Nó sử dụng một yêu cầu HTTP POST để gửi một thực thể đơn hàng đến máy chủ, và phản hồi được sử dụng để xác định thành công hoặc thất bại của hoạt động.
* Phương thức dispose: Phương thức dispose được ghi đè để dọn dẹp tài nguyên, chẳng hạn như việc loại bỏ \_quantityController.
* Phương thức build: Phương thức build được ghi đè để định nghĩa giao diện người dùng cho màn hình. Nó bao gồm một Scaffold với thanh ứng dụng, chi tiết thực phẩm, một trường văn bản để nhập số lượng, chi tiết giá tổng cộng, một hình ảnh banner và một nút thanh toán. Phương thức buildFoodDetails: Phương thức này hiển thị chi tiết thực phẩm như tên, thành phần và giá. Nó sử dụng hiển thị điều kiện dựa trên trạng thái tải.
* Phương thức buildTotalPrice: Phương thức này hiển thị chi tiết giá tổng cộng, bao gồm giá cơ bản, số lượng, phí vận chuyển và tổng cộng chung.
* Phương thức buildPriceRow: Một phương thức hỗ trợ để xây dựng một dòng để hiển thị một chi tiết liên quan đến giá cụ thể.
* Phương thức buildPaymentButton: Phương thức này hiển thị một nút thanh toán mà kích hoạt phương thức pushData khi được chạm vào. Giao diện của nút phụ thuộc vào trạng thái tải.

# Món Ăn Đã Đặt

* A screenshot of a phone

  Description automatically generatedImports: Các gói và tệp tin cần thiết được nhập vào. Điều này bao gồm Flutter, HTTP requests, shared preferences và các widgets được sử dụng trong ứng dụng.
* Widget Definition: Lớp OrderHistoryScreen được định nghĩa là một StatefulWidget, không chấp nhận tham số.
* Lớp State: Lớp \_OrderHistoryScreenState mở rộng từ State và chứa logic để hiển thị lịch sử đơn hàng, bao gồm cả việc tải dữ liệu từ máy chủ.
* Khởi tạo Trạng thái: Phương thức initState được ghi đè để khởi tạo trạng thái khi widget được tạo. Nó gọi phương thức fetchData để tải dữ liệu đơn hàng từ máy chủ.
* Phương thức fetchData: Phương thức này thực hiện một yêu cầu HTTP GET để lấy danh sách đơn hàng dựa trên idUser từ shared preferences. Dữ liệu nhận được được sử dụng để cập nhật trạng thái.
* Phương thức build: Phương thức build được ghi đè để định nghĩa giao diện người dùng cho màn hình. Nó bao gồm một Scaffold với thanh ứng dụng, danh sách đơn hàng và thanh điều hướng dưới cùng.
* Widget BottomNavigationBar: Sử dụng widget BottomBarWidget để tạo thanh điều hướng dưới cùng với khả năng chuyển đổi giữa các tab.
* Phương thức buildBottomNavigationBar: Phương thức này xây dựng thanh điều hướng dưới cùng và truyền hàm changeTab để xử lý sự kiện chuyển đổi tab.
* Phương thức buildBody: Phương thức này xây dựng phần thân của màn hình, hiển thị trạng thái tải, thông báo lỗi hoặc danh sách đơn hàng.
* ListView.builder: Sử dụng ListView.builder để hiển thị danh sách đơn hàng. Mỗi đơn hàng được hiển thị bằng cách sử dụng ListOrderHistoryWidget.
* Phương thức changeTab: Phương thức này được gọi khi người dùng chuyển đổi giữa các tab dưới thanh điều hướng dưới cùng.

# Thông Báo

* Imports: Các gói và tệp tin cần thiết được nhập vào, bao gồm Flutter và widget BottomBarWidget.
* Widget Definition: Lớp NotificationScreen được định nghĩa là một StatefulWidget, không chấp nhận tham số.
* Lớp State: Lớp \_NotificationScreenState mở rộng từ State và chứa logic để hiển thị màn hình thông báo, bao gồm cả việc chuyển đổi giữa các tab dưới thanh điều hướng.
* Khởi tạo Trạng thái: Phương thức initState không được sử dụng trong đoạn mã này.
* Phương thức build: Phương thức này định nghĩa giao diện người dùng cho màn hình. Nó bao gồm một Scaffold với thanh ứng dụng, danh sách thông báo và thanh điều hướng dưới cùng.
* A screenshot of a cell phone

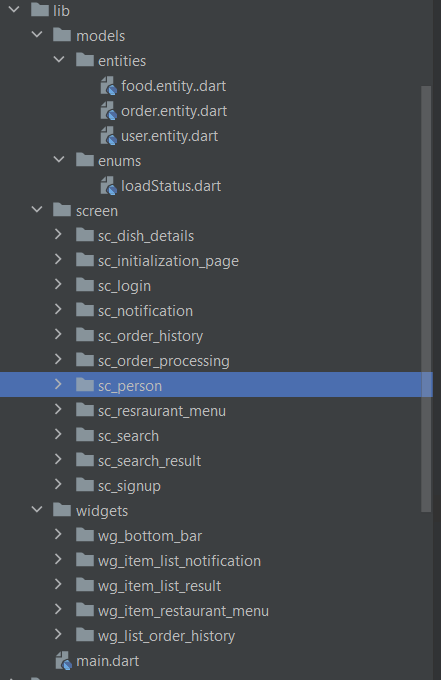
  Description automatically generatedListView: Sử dụng ListView để hiển thị danh sách các thông báo. Mỗi thông báo được hiển thị bằng cách sử dụng widget ItemListNotificationWidget.
* Phương thức ItemListNotificationWidget: Phương thức này trả về một widget Padding chứa một mục thông báo. Mỗi mục bao gồm hình ảnh, tiêu đề, nội dung và thời gian.
* Widget Ink: Sử dụng Ink để tạo một tác động mờ khi người dùng chạm vào mục thông báo.
* Padding và Column: Sử dụng Padding và Column để định dạng và sắp xếp các phần tử bên trong mục thông báo.
* Image.asset: Hiển thị hình ảnh người dùng trong mỗi mục thông báo.
* Text Widgets: Sử dụng Text để hiển thị tiêu đề, nội dung và thời gian của mỗi thông báo.
* Divider: Sử dụng Divider để tạo đường phân cách giữa các mục thông báo.
* BottomBarWidget: Sử dụng BottomBarWidget để hiển thị thanh điều hướng dưới cùng với khả năng chuyển đổi giữa các tab.

# Thông Tin Người Dùng

* A screenshot of a phone

  Description automatically generatedImports: Import các gói và tệp tin cần thiết, bao gồm Flutter, Dotenv, các enums và widgets khác.
* Widget Definition: Lớp PersonScreen được định nghĩa là một StatefulWidget, không chấp nhận tham số.
* Lớp State: Lớp \_PersonScreenState mở rộng từ State và chứa logic để hiển thị màn hình thông tin cá nhân, bao gồm cả việc tải dữ liệu từ máy chủ.
* Khởi tạo Trạng thái: Phương thức initState được gọi khi widget được tạo để thực hiện việc tải dữ liệu người dùng.
* Phương thức fetchData: Phương thức này thực hiện một yêu cầu HTTP GET để lấy thông tin cá nhân của người dùng từ máy chủ.
* Phương thức build: Phương thức build định nghĩa giao diện người dùng cho màn hình. Nó sử dụng một Scaffold chứa thông tin cá nhân, lịch sử đơn hàng, và một thanh điều hướng dưới cùng.
* Widget CircleAvatar và Background Image: Hiển thị hình ảnh đại diện và nền của màn hình thông tin cá nhân.
* Phần Thông tin Cá nhân: Sử dụng một Card để hiển thị thông tin cá nhân của người dùng, bao gồm ID, số điện thoại và địa chỉ.
* Divider: Sử dụng Divider để tạo đường phân cách giữa các phần. InkWell và Mục Lịch sử Đơn hàng: Sử dụng InkWell để thêm tương tác khi nhấp vào mục lịch sử đơn hàng, hiển thị thông tin và chuyển hướng sang màn hình lịch sử đơn hàng.
* Nút Đăng xuất: Sử dụng một ElevatedButton để đăng xuất người dùng khỏi ứng dụng và chuyển hướng đến màn hình đăng nhập. BottomBarWidget: Sử dụng BottomBarWidget để hiển thị thanh điều hướng dưới cùng và có khả năng chuyển đổi giữa các tab.
* Phương thức changeTab: Phương thức này được gọi khi người dùng chuyển đổi giữa các tab trên thanh điều hướng dưới cùng.

4. Cấu trúc sản phẩm



5. Tổng kết

* Môi Trường Lập Trình:
* Sử dụng ngôn ngữ Dart và framework Flutter để phát triển ứng dụng di động đa nền tảng.
* Sử dụng các công cụ hỗ trợ như Visual Studio Code và các gói Dart để tối ưu hóa quá trình lập trình.
* Tương Tác Mạng:
* Sử dụng thư viện HTTP của Dart để tương tác với API và truy xuất dữ liệu từ máy chủ.
* Xử lý các trạng thái tải dữ liệu bằng cách sử dụng enum LoadStatus để hiển thị giao diện người dùng thích hợp.
* Quản Lý Trạng Thái Ứng Dụng:
* Sử dụng StatefulWidget và State để quản lý trạng thái của các widget và cập nhật giao diện người dùng khi có thay đổi.
* Định Dạng Giao Diện:
* Sử dụng các widget như Container, ListView, Column, và Row để xây dựng giao diện người dùng đa dạng và thân thiện với người dùng.
* ***Cuối cùng ứng dụng di động đang trở thành một phần quan trọng không thể thiếu trong cuộc sống hàng ngày của chúng ta, và sự phổ biến của Flutter đã mang lại một sự đổi mới đáng kể trong quá trình phát triển ứng dụng di động. Trong báo cáo này, chúng ta đã xem xét quá trình lập trình ứng dụng di động bằng ngôn ngữ Flutter, một framework đa nền tảng do Google phát triển.***