

Widget Layout

...

Layout

- Trong Flutter, chúng ta chỉ có Widget và sắp xếp chúng
- Tất cả các Widget cần được bọc bởi một Widget lớn hơn có tính tổ chức
- Layout trong Flutter chính là bao gồm một hệ thống phân cấp các Widget và ràng buộc chúng với nhau

Điểm chung của các loại Layout

- Các Widget layout thường sẽ chứa các Widget con trong tham số `child` ở constructor đối với các layout chỉ chứa một Widget duy nhất, như `Container`. Hay `children` đối với các Layout có thể chứa nhiều Widget con như `Column` và `Row`.
- Đối với `children`, thì giá trị của nó là một danh sách các Widget

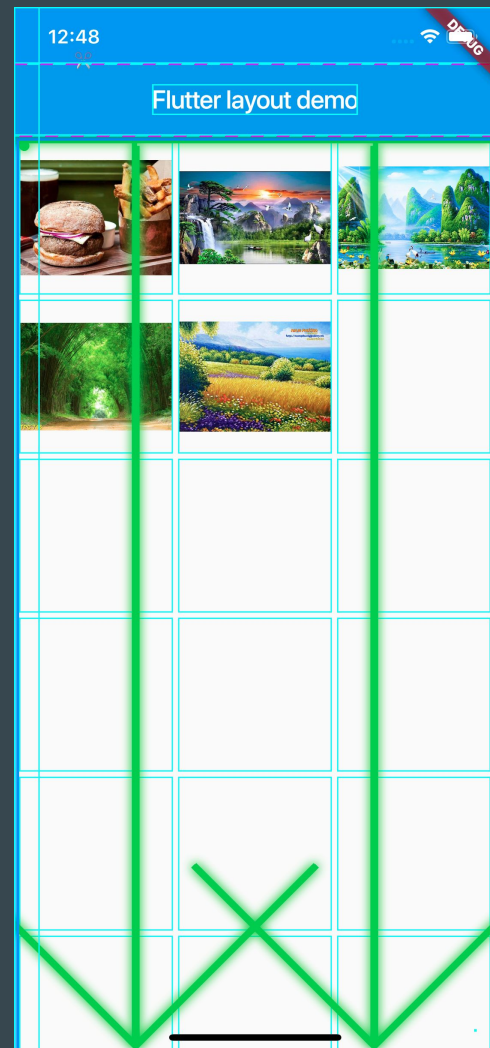
Nguyên tắc layout

Trước khi dựng một màn hình cần tính toán và dựng sơ đồ layout, chúng ta phải chia nhỏ được các thành phần:

- Xác định hàng và cột và các khối
- Xác định có cần sử dụng List
- Xác định có chứa các yếu tố chồng chéo không
- Xác định các thành phần cần đặt tỉ lệ với nhau

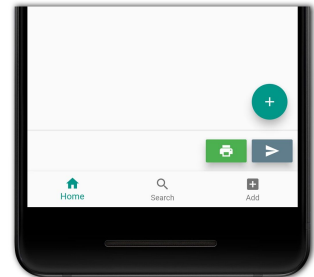
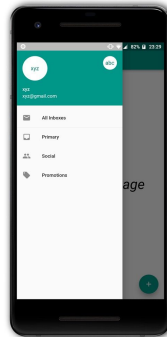
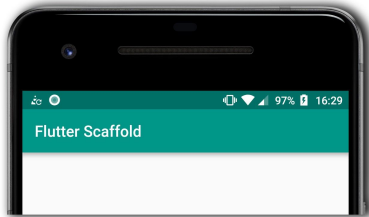
Virtual Rendering

```
debugPaintSizeEnabled = true
```



Scaffold

- Scaffold: Widget này được cung cấp bởi material package. Nó cung cấp một cách dễ dàng để thêm AppBar, FloatingActionButton, Drawer, bottomNavigationBar, SnackBar, v.v.



Container

- Có thể coi **container** là một Widget Layout đơn giản, chỉ chứa đựng đúng **một Widget con**.
- Container đang là widget duy nhất có các thuộc tính margin, padding, constraint, width, height được chỉ định trong constructor

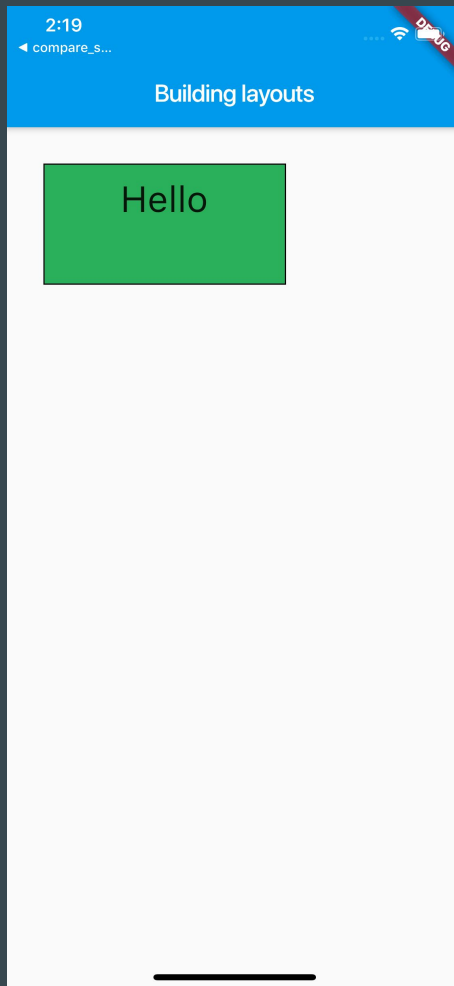
Lưu ý với Container

- Lưu ý:
- + Nếu container không có các thuộc tính định vị với bên ngoài như margin, constraint hay bên trong như padding, và cũng không được định width, height thì khi render, kích thước của nó sẽ nhỏ nhất có thể
- + Nếu container có thuộc tính constraint theo parent Widget chứa nó, thì nó sẽ dẫn ra cho khớp constraints được định.
- + Nếu sử dụng quá nhiều container có thể gây cảm giác giật, lag trên thiết bị có cấu hình yếu, vì nó sẽ mất khá nhiều thời gian tính toán so với các loại layout khác


```
Container(  
  color: Colors.green,  
),
```



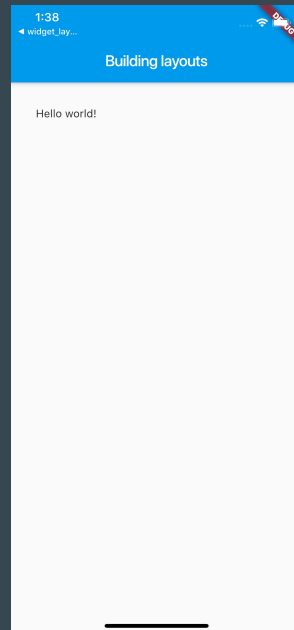
```
Widget myLayoutWidget() {  
  return Container(  
    margin: EdgeInsets.all(30.0),  
    padding: EdgeInsets.all(10.0),  
    alignment: Alignment.topCenter,  
    width: 200,  
    height: 100,  
    decoration: BoxDecoration(  
      color: Colors.green,  
      border: Border.all(),  
    ),  
    child: Text("Hello", style: TextStyle(fontSize: 30)),  
  );  
}
```



Padding

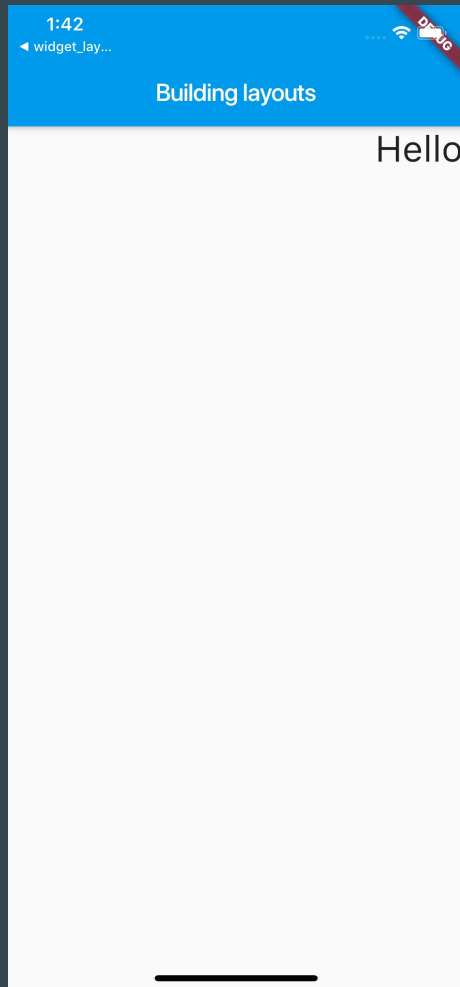
- Param của padding là một instance của `EdgeInsetsGeometry`
- Padding sẽ định nghĩa ra `paddingLeft`, `paddingTop`, `paddingRight`, `paddingBottom`

```
Widget myLayoutWidget() {  
    return Padding(  
        padding: EdgeInsets.all(32.0),  
        child: Text("Hello world!"),  
    );  
}
```



Align

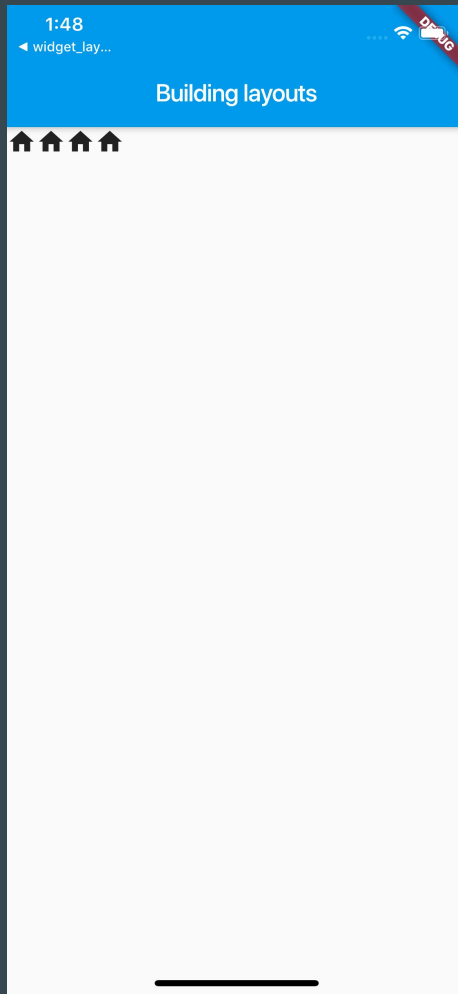
```
Widget myLayoutWidget() {  
  return Align(  
    alignment: Alignment.topRight,  
    child: Text(  
      "Hello",  
      style: TextStyle(fontSize: 30),  
    ),  
  );  
}
```



Row

- Row dùng để hiển thị các widget con theo cách nằm ngang

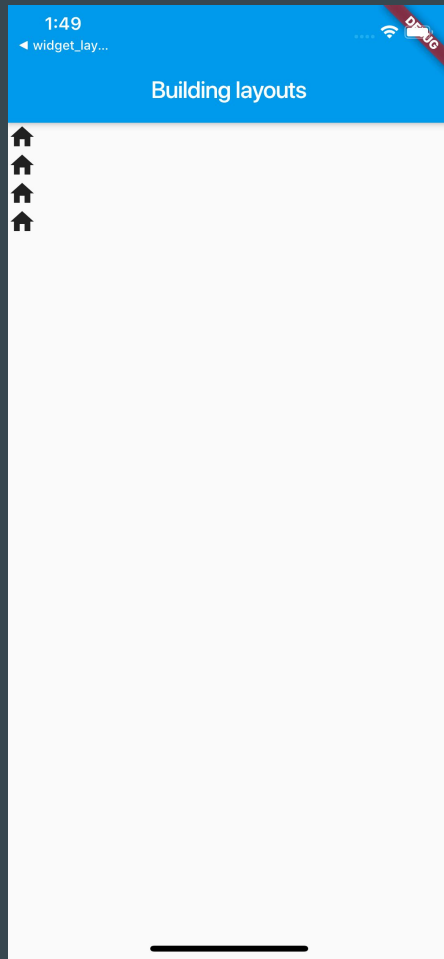
```
Widget myLayoutWidget() {  
  return Row(  
    children: [  
      Icon(Icons.home),  
      Icon(Icons.home),  
      Icon(Icons.home),  
      Icon(Icons.home),  
    ],  
  );  
}
```



Column

- Column dùng để hiển thị các widget con theo hàng dọc

```
Widget myLayoutWidget() {  
  return Column(  
    children: [  
      Icon(Icons.home),  
      Icon(Icons.home),  
      Icon(Icons.home),  
      Icon(Icons.home),  
    ],  
  );  
}
```



CrossAxisAlignment của Row và Column

- Sử dụng CrossAxisAlignment để đặt column và row theo hướng mong muốn.

Ví dụ:

`CrossAxisAlignment.start`

`CrossAxisAlignment.center`

`CrossAxisAlignment.end`

`CrossAxisAlignment.stretch`

`CrossAxisAlignment.baseline`

TextDirection của Row và Column

- Sử dụng TextDirection để sắp xếp các Widget theo chiều ngang và cách diễn giải bắt đầu và kết thúc theo hướng ngang.

TextDirection.rtl

TextDirection.ltr

VerticalDirection của Row và Column

- Sử dụng VerticalDirection để sắp xếp các Widget theo chiều dọc và cách diễn giải bắt đầu và kết thúc theo chiều dọc.

VerticalDirection.down

VerticalDirection.up

MainAxisAlignment của Row và Column

- Sử dụng MainAxisAlignment để sắp xếp vị trí các widget con trên trục chính

MainAxisAlignment.start

MainAxisAlignment.center

MainAxisAlignment.end

MainAxisAlignment.spaceAround

MainAxisAlignment.spaceBetween

MainAxisAlignment.spaceEvenly

MainAxisSize của Row và Column

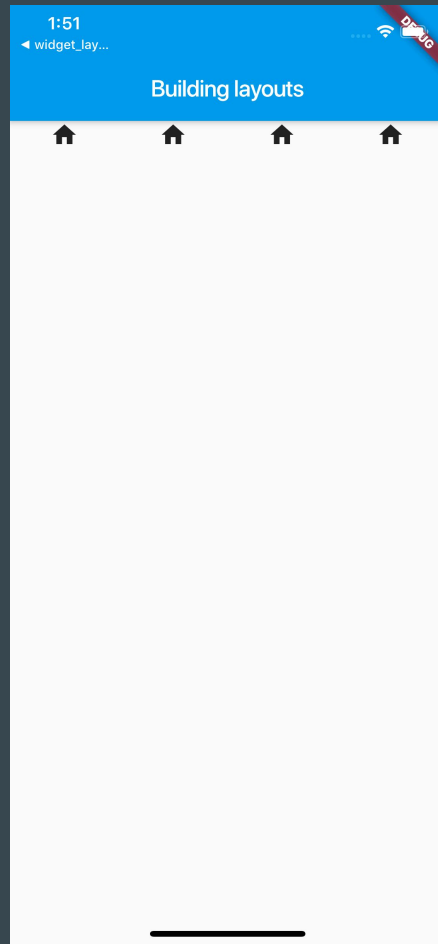
- Thuộc tính chỉ kích thước nên được phân bổ trên trục chính

`MainAxisSize.min`

`MainAxisSize.max`

Expanded

```
Widget myLayoutWidget() {  
  return Row(  
    children: [  
      Expanded(child: Icon(Icons.home)),  
      Expanded(child: Icon(Icons.home)),  
      Expanded(child: Icon(Icons.home)),  
      Expanded(child: Icon(Icons.home)),  
    ],  
  );  
}
```

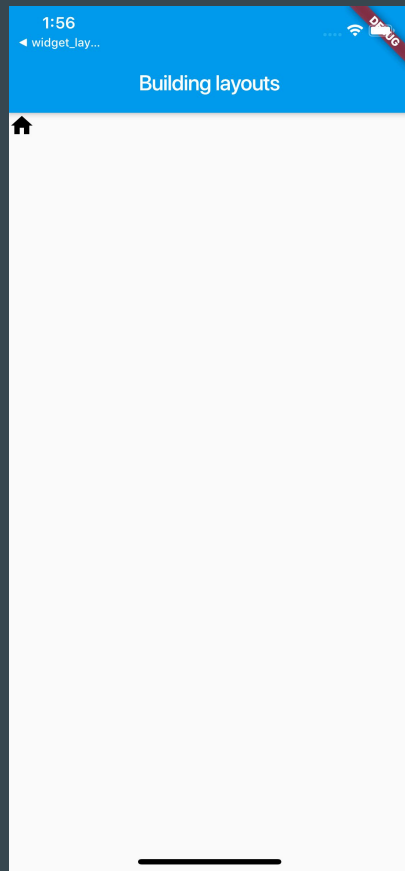


```
Widget myLayoutWidget() {  
  return Row(  
    children: [  
      Expanded(  
        flex: 7,  
        child: Container(  
          color: Colors.green,  
        ),  
      ),  
      Expanded(  
        flex: 3,  
        child: Container(  
          color: Colors.yellow,  
        ),  
      ),  
    ],  
  );  
}
```

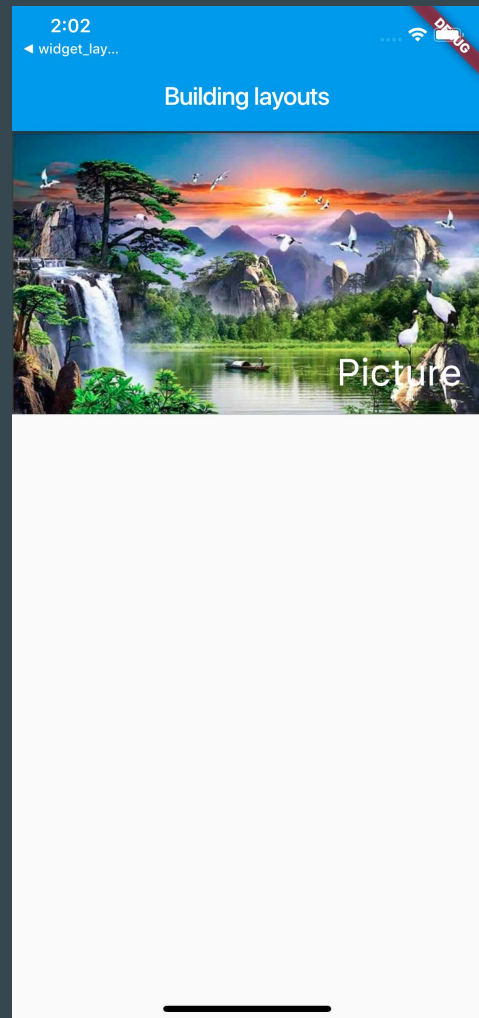


Stacks

```
Widget myLayoutWidget() {  
  return Stack(  
    children: [  
      Icon(Icons.home),  
      Icon(Icons.home),  
      Icon(Icons.home),  
      Icon(Icons.home),  
    ],  
  );  
}
```

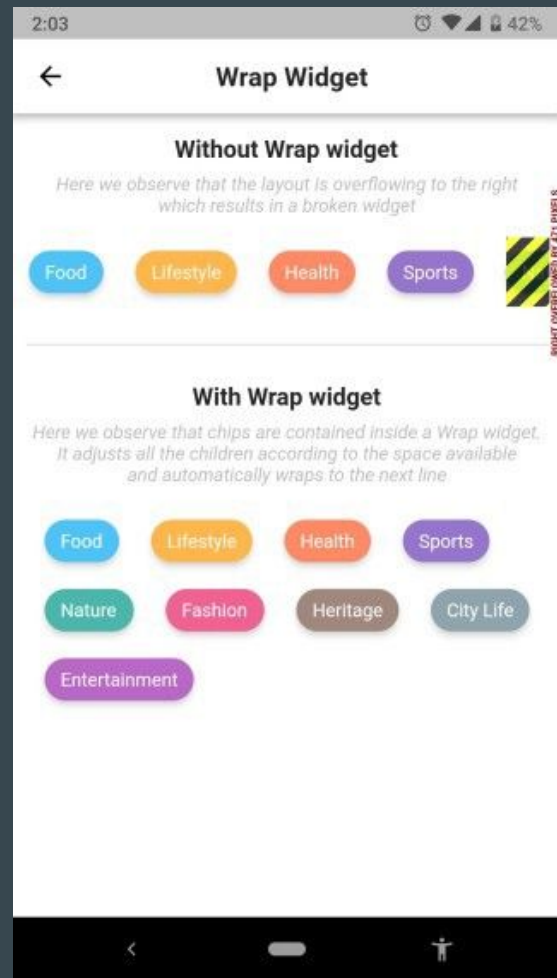


```
Widget myLayoutWidget() {  
  return Stack(  
    alignment: Alignment.bottomRight,  
    children: [  
      Image.asset('images/1.jpg'),  
      Padding(  
        padding: EdgeInsets.all(16.0),  
        child: Text(  
          'Picture',  
          style: TextStyle(fontSize: 30, color: Colors.white),  
        ),  
      ),  
    ],  
  );  
}
```



Wrap

- Wrap tương tự với Row và Column.
- Ưu điểm của nó là có thể điều chỉnh widget con theo không gian có sẵn trên màn hình
- Mặc định hiển thị của wrap là ngang, tuy nhiên có thể custom thành dọc



ListView

- ListView: Widget này cuộn các hàng hoặc cột với nội dung lớn hơn kích thước vật lý của màn hình
- List cho phép cuộn theo chiều dọc hoặc chiều ngang
- Các loại ListView thường dùng:

ListView

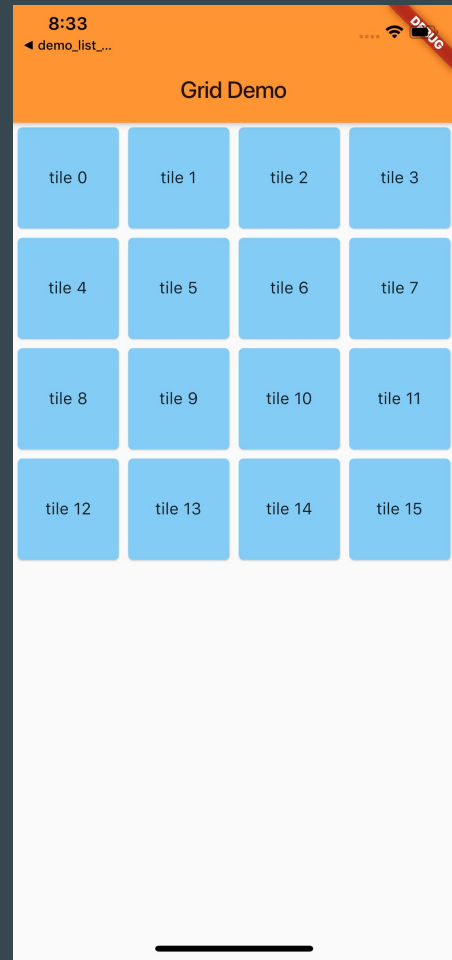
ListView.builder

ListView.separated

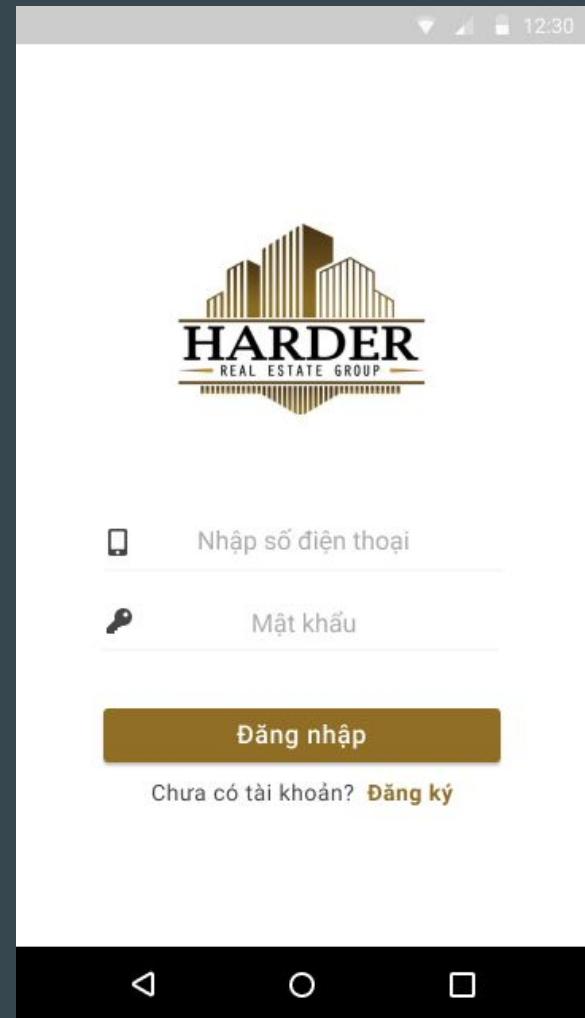
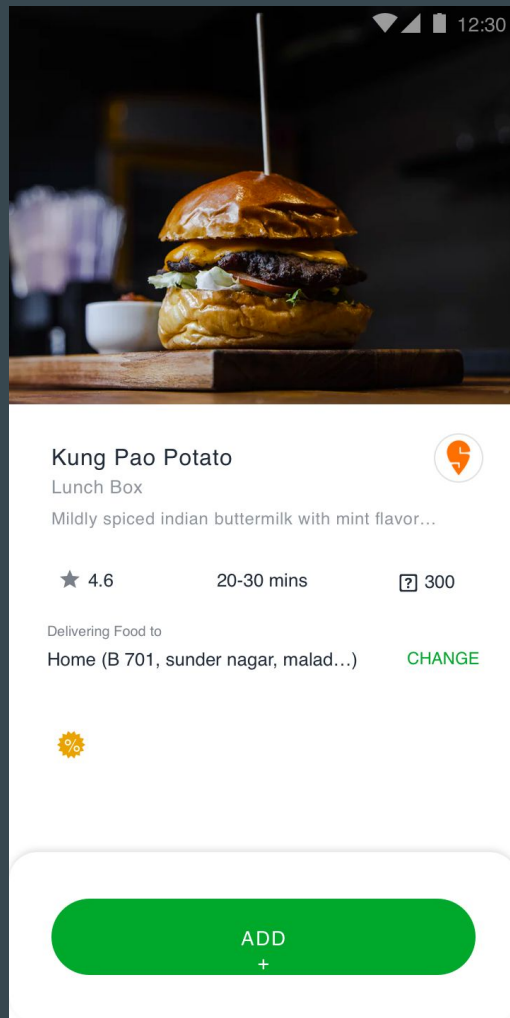
ListView.custom

GridView

- GridView: Widget này cuộn nội dung được đặt trong một lưới các hàng và cột.




Bài tập:



Bài tập về nhà:

12:30



Họ và tên

Số điện thoại

Mật khẩu

Nhập lại mật khẩu


Đăng ký

Đã có tài khoản? **Đăng nhập**

12:30

Hồ sơ

Cập nhật



Đổi ảnh đại diện

Họ và tên
Hàn Duy

Số điện thoại
0902 209 011

Địa chỉ
14 gõ 6 Phạm Tuấn Tài - Quận Tân Phú - Thành phố Hồ Chí Minh