代码清单

——基于Flutter的移动端跨平台记账APP



项目名称： 账单记录APP

专业班级： 软件工程1802

小组成员：徐任、牟灵成、莫丁阳

指导教师： 杨枨老师

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 备注 |
| 01 | 2020.12.15 | 0.1 | 徐任 | 初稿 |
| 02 | 2020.12.20 | 0.2 | 徐任 |  |
| 03 | 2020.12.28 | 0.3 | 徐任 |  |

# 目录

[代码清单 1](#_Toc60098047)

[目录 2](#_Toc60098048)

[1引言 3](#_Toc60098049)

[1.1标识 3](#_Toc60098050)

[1.2系统概述 3](#_Toc60098051)

[1.3文档概述 4](#_Toc60098052)

[2代码清单 4](#_Toc60098053)

[2.1 add 4](#_Toc60098054)

[2.2 addIncome 5](#_Toc60098055)

[2.3 addPay 7](#_Toc60098056)

[2.4 billDetail 10](#_Toc60098057)

[2.5 changeIncome 16](#_Toc60098058)

[2.6 changePay 19](#_Toc60098059)

[2.7 chartIncome 22](#_Toc60098060)

[2.8 chartIncomeMouth 23](#_Toc60098061)

[2.9 chartIncomeYear 30](#_Toc60098062)

[2.10 chartPay 37](#_Toc60098063)

[2.11 chartPayMouth 38](#_Toc60098064)

[2.12 chartPayYear 44](#_Toc60098065)

[2.13 charts 51](#_Toc60098066)

[2.14 details 52](#_Toc60098067)

[2.15 forgetPwd 60](#_Toc60098068)

[2.16 login 64](#_Toc60098069)

[2.17 modifyPwd 69](#_Toc60098070)

[2.18 person 73](#_Toc60098071)

[2.19 register 85](#_Toc60098072)

[2.20 bill 92](#_Toc60098073)

[2.21 numberKeyboard 93](#_Toc60098074)

[2.22 routes 101](#_Toc60098075)

[2.23 tabs 102](#_Toc60098076)

[2.24 types 103](#_Toc60098077)

[2.25 pubspec 105](#_Toc60098078)

[2.26 rank 107](#_Toc60098079)

# 1引言

## 1.1标识

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | SE2020-G14-代码清单 |
| 当前版本： | 0.3 |
| 作者： | 徐任、牟灵成、莫丁阳 |
| 完成日期： | 2020-12-28 |

## 1.2系统概述

本文档适用账单记录APP项目：“制账”。

“制账”是一款以记录账单、展示账单为主要功能的清洁 APP，追求简洁界面、充实功能、少广告、无收费，操作简单易上手，图表分析易看懂。

需方：杨枨老师；

投资方兼开发方：G14小组；

当然用户：杨枨老师；

典型用户：邢海粟、廖安琪、赵宇阳；

支持机构：浙大城市学院；

项目首先会在G14小组成员电脑上的模拟器试运行，正式版将在杨枨老师的手机上运行。

开发历史：

2020.9.29-2020.9.30 确定选题

2020.10.1-2020.10.4 项目介绍初稿

2020.10.10-2020.10.11 项目介绍修改

2020.10.13-2020.10.18 项目介绍终稿

2020.10.20-2020.10.25 项目计划及PPT制作

2020.10.27-2020.11.1 项目计划修订

2020.11.3-2020.11.8 可行性分析

2020.11.10-2020.11.15 软件需求规格说明初稿

2020.11.17-2020.11.22 软件需求规格说明修订

2020.11.24-2020.11.26 总体设计说明初稿

2020.11.26-2020.11.29 总体设计说明修订

2020.12.1-2020.12.6 详细设计说明初稿

2020.12.8-2020.12.13 详细设计说明修订

相关文档：

SE2020-G14-正式选题1.0.docx

SE2020-G14-项目计划书1.4.docx

G14版本命名规则1.0.docx

SE2020-G14-可行性分析(研究)报告(FAR)1.2.docx

SE2020-G14-软件需求说明(SRS)1.2.docx

SE2020-G14-详细设计说明1.0.docx

## 1.3文档概述

本项目开发总结报告用于指导开发“制账”APP项目顺利进行并最终通过评审的项目产品。本项目开发总结报告面向项目组全体成员。

# 2代码清单

## 2.1 add

import 'package:flutter/material.dart';

import 'package:zhizhang/pages/addPay.dart';

import 'package:zhizhang/pages/addIncome.dart';

class AddPage extends StatefulWidget {

@override

AddPageState createState() => AddPageState();

}

class AddPageState extends State<AddPage> {

@override

Widget build(BuildContext context) {

return DefaultTabController(

length: 2,

child: Scaffold(

//头部

appBar: AppBar(

iconTheme: IconThemeData(

color: Colors.black,

), //修改颜色

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

title: Row(

children: <Widget>[

Expanded(

child: TabBar(

labelColor: Colors.black,

indicatorColor: Colors.black,

indicatorSize: TabBarIndicatorSize.tab,

tabs: [

Tab(text: "支出"),

Tab(text: "收入")

])) //头部导航

],

),

),

body: TabBarView(

children: [AddPayPage() /\*支出增加 \*/, AddIncomePage() /\*收入增加 \*/], //显示页面

),

));

}

}

## 2.2 addIncome

import 'package:flutter/material.dart';

import 'package:zhizhang/utils/bill.dart';

import 'package:zhizhang/utils/numberKeyboard.dart';

import 'package:zhizhang/utils/types.dart';

class AddIncomePage extends StatefulWidget {

@override

AddIncomePageState createState() => AddIncomePageState();

}

class AddIncomePageState extends State<AddIncomePage> {

var color = [false, false, false]; //判断按钮是否被选中

//改变按钮颜色

void \_set(int index) {

var value = color.indexWhere((element) => element == true);

if (value != -1) {

color[value] = false; //恢复之前按钮颜色

}

color[index] = true; //改变选中按钮颜色

}

//按钮布局及事件

flatButton(int index) {

return FlatButton(

color: Colors.white,

highlightColor: Colors.white,

splashColor: Colors.white,

onPressed: () {

setState(() {

\_set(index); //改变按钮颜色

});

//显示数字键盘

showModalBottomSheet(

context: context,

builder: (context) {

return Container(

child: NumberKeyboard(

bill: Bill(0, index),

action: 0,

),

);

});

},

child: Column(

children: <Widget>[

Container(

color: null,

child: CircleAvatar(

backgroundColor: color[index]

? Color.fromARGB(255, 255, 200, 0)

: Color.fromARGB(255, 239, 239, 239), //按钮颜色

radius: 30.0,

child: Icon(incomeIcons[index], color: Colors.black38), //按钮图标

),

),

Text(incomeTypes[index]) //按钮label

],

),

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: Container(

color: Colors.white,

child: GridView.count(

crossAxisSpacing: 8,

mainAxisSpacing: 10,

crossAxisCount: 3,

padding: EdgeInsets.all(20),

children: [flatButton(0), flatButton(1), flatButton(2)], //增加按钮

),

));

}

}

## 2.3 addPay

import 'package:flutter/material.dart';

import 'package:zhizhang/utils/bill.dart';

import 'package:zhizhang/utils/types.dart';

import 'package:zhizhang/utils/numberKeyboard.dart';

class AddPayPage extends StatefulWidget {

@override

AddPayPageState createState() => AddPayPageState();

}

class AddPayPageState extends State<AddPayPage> {

var price = TextEditingController();

var color = [

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false

]; //判断按钮是否被选中

//数字键盘

//改变按钮颜色

void \_set(int index) {

var value = color.indexWhere((element) => element == true);

if (value != -1) {

color[value] = false; //恢复之前按钮颜色

}

color[index] = true; //改变选中按钮颜色

}

//按钮布局及事件

flatButton(int index) {

return FlatButton(

color: Colors.white,

highlightColor: Colors.white,

splashColor: Colors.white,

onPressed: () {

setState(() {

\_set(index); //改变按钮颜色

});

//显示数字键盘

showModalBottomSheet(

context: context,

builder: (context) {

return Container(

child: NumberKeyboard(

bill: Bill(1, index),

action: 0,

),

);

});

},

child: Column(

children: <Widget>[

Container(

color: null,

child: CircleAvatar(

backgroundColor: color[index]

? Color.fromARGB(255, 255, 200, 0)

: Color.fromARGB(255, 239, 239, 239), //按钮颜色

radius: 30.0,

child: Icon(payIcons[index], color: Colors.black38), //按钮图标

),

),

Text(payTypes[index]) //按钮label

],

),

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: Container(

color: Colors.white,

//页面按钮布局

child: GridView.count(

crossAxisSpacing: 8,

mainAxisSpacing: 10,

crossAxisCount: 3,

padding: EdgeInsets.all(20),

children: <Widget>[

flatButton(0),

flatButton(1),

flatButton(2),

flatButton(3),

flatButton(4),

flatButton(5),

flatButton(6),

flatButton(7),

flatButton(8),

flatButton(9),

flatButton(10),

flatButton(11),

flatButton(12),

flatButton(13),

flatButton(14),

flatButton(15),

flatButton(16),

flatButton(17),

flatButton(18),

], //增加按钮

)));

}

}

## 2.4 billDetail

import 'package:flutter/material.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

//单个账单明细

// ignore: must\_be\_immutable

class BillDatilPage extends StatefulWidget {

var arguments;

BillDatilPage({Key key, this.arguments}) : super(key: key);

BillDatilPageState createState() =>

BillDatilPageState(

arguments: this.arguments

);

}

class BillDatilPageState extends State<BillDatilPage> {

int billId;

int billTypesId;

int billFromId;

double billAmount;

int billYear;

int billMonth;

int billDay;

int billHour;

String billRemark; //上述属性为账单属性

var arguments;

BillDatilPageState({this.arguments}); //总体账单页面传值

//删除

\_delete() async {

var conn = await MySqlConnection.connect(connect);

await conn

.prepared('Delete from bill where billId = ?', [arguments["billId"]]);

conn.close();

}

//根据传值对账单属性赋值

@override

initState() {

super.initState();

billId = arguments['billId'];

billFromId = arguments['billFromId'];

billTypesId = arguments['billTypesId'];

billAmount = arguments['billAmount'];

billYear = arguments['billYear'];

billMonth = arguments['billMonth'];

billDay = arguments['billDay'];

billHour = arguments['billHour'];

billRemark = arguments['billRemark'];

}

//刷新页面

\_refresh() async {

var conn = await MySqlConnection.connect(connect);

Results result = await (await conn.execute(

'select \* from bill where billId = ${arguments["billId"]}'))

.deStream();

//更新账单属性

setState(() {

billTypesId = result.map((v) => v.byName('billTypesId')).single;

billAmount = result.map((v) => v.byName('billAmount')).single;

billYear = result.map((v) => v.byName('billYear')).single;

billMonth = result.map((v) => v.byName('billMonth')).single;

billDay = result.map((v) => v.byName('billDay')).single;

billHour = result.map((v) => v.byName('billHour')).single;

billRemark = result.map((v) => v.byName('billRemark')).single;

});

conn.close();

}

@override

Widget build(BuildContext context) {

return Scaffold(

//头部

appBar: PreferredSize(

child: AppBar(

leading: IconButton(

icon: Icon(Icons.keyboard\_backspace),

onPressed: () {

Navigator.of(context).pop(true);

},

), //返回总体账单页面

iconTheme: IconThemeData(

color: Colors.black,

), //修改图标颜色

backgroundColor: Color.fromARGB(255, 255, 200, 0), //背景色

title: Column(

children: <Widget>[

Icon(

billFromId == 1

? payIcons[billTypesId]

: incomeIcons[billTypesId],

size: 35,

color: Colors.white,

), //账单图标

Text(billFromId == 1

? payTypes[billTypesId]

: incomeTypes[billTypesId]) //账单类型

],

),

centerTitle: true, //居中

),

preferredSize: Size(MediaQuery.of(context).size.width,

MediaQuery.of(context).size.height / 10)),

body: Column(

children: <Widget>[

//类型

Container(

padding: EdgeInsets.all(20.0),

child: Row(children: <Widget>[

Text("类型", style: TextStyle(color: Colors.grey, fontSize: 20)),

SizedBox(

width: 10,

),

Text(billFromId == 1 ? "支出" : "收入",

style: TextStyle(fontSize: 20))

]),

),

//分隔线

Container(

width: MediaQuery.of(context).size.width,

height: 0.5,

color: Colors.grey,

),

//金额

Container(

padding: EdgeInsets.all(20.0),

child: Row(children: <Widget>[

Text("金额", style: TextStyle(color: Colors.grey, fontSize: 20)),

SizedBox(

width: 10,

),

Text("$billAmount", style: TextStyle(fontSize: 20))

]),

),

//分隔线

Container(

width: MediaQuery.of(context).size.width,

height: 0.5,

color: Colors.grey,

),

//时间

Container(

padding: EdgeInsets.all(20.0),

child: Row(children: <Widget>[

Text("时间", style: TextStyle(color: Colors.grey, fontSize: 20)),

SizedBox(

width: 10,

),

Text("$billYear 年 $billMonth 月 $billDay 日 $billHour 时",

style: TextStyle(fontSize: 20))

]),

),

//分隔线

Container(

width: MediaQuery.of(context).size.width,

height: 0.5,

color: Colors.grey,

),

//备注

Container(

padding: EdgeInsets.all(20.0),

child: Row(children: <Widget>[

Text("备注", style: TextStyle(color: Colors.grey, fontSize: 20)),

SizedBox(

width: 10,

),

Text(billRemark, style: TextStyle(fontSize: 20))

]),

),

//分隔线

Container(

width: MediaQuery.of(context).size.width,

height: 0.5,

color: Colors.grey,

),

SizedBox(

height: MediaQuery.of(context).size.height / 2.4,

),

//按钮

Container(

width: MediaQuery.of(context).size.width,

child: Row(children: <Widget>[

//编辑

Container(

width: MediaQuery.of(context).size.width / 2,

child: FlatButton(

onPressed: () {

\_change(); //跳转至修改页面

},

child: Text("编辑")),

),

//分隔线

Container(

width: 0.5,

height: 20,

color: Colors.grey,

),

//删除

Container(

width: MediaQuery.of(context).size.width / 2 - 0.5,

child: FlatButton(

onPressed: () {

//提示

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("确认删除"),

content: Text("删除后数据不可恢复！"),

actions: <Widget>[

FlatButton(

child: Text("取消",

style: TextStyle(

color: Color.fromARGB(

255, 255, 200, 0))),

onPressed: () {

Navigator.of(context)

.pop(true); //取消删除，返回明细页面

}),

FlatButton(

child: Text("删除",

style: TextStyle(

color:

Color.fromARGB(255, 255, 200, 0))),

onPressed: () async {

await \_delete();

Navigator.of(context).pop();

Navigator.of(context).pop(true); //删除，并返回账单页面

},

)

],

);

},

);

},

child: Text("删除")),

)

]),

)

],

),

);

}

//根据账单类型跳转至“支出”或“收入”修改

\_change() {

if (arguments["billFromId"] == 1) {

Navigator.pushNamed(context, "/changePay", arguments: {

"billId": billId,

"billFromId": billFromId,

"billTypesId": billTypesId,

"billAmount": billAmount,

"billYear": billYear,

"billMonth": billMonth,

"billDay": billDay,

"billHour": billHour,

"billRemark": billRemark

}).then((value) => {

if (value == true) {\_refresh()}

}); //如果修改，刷新页面

} else {

Navigator.pushNamed(context, "/changeIncome", arguments: {

"billId": billId,

"billFromId": billFromId,

"billTypesId": billTypesId,

"billAmount": billAmount,

"billYear": billYear,

"billMonth": billMonth,

"billDay": billDay,

"billHour": billHour,

"billRemark": billRemark

}).then((value) => {

if (value == true) {\_refresh()}

}); //如果修改，刷新页面

}

}

}

## 2.5 changeIncome

import 'package:flutter/material.dart';

import 'package:zhizhang/utils/bill.dart';

import 'package:zhizhang/utils/numberKeyboard.dart';

import 'package:zhizhang/utils/types.dart';

// ignore: must\_be\_immutable

class ChangeIncomePage extends StatefulWidget {

var arguments;

ChangeIncomePage({Key key, this.arguments}) : super(key: key);

@override

ChangeIncomePageState createState() =>

ChangeIncomePageState(arguments: this.arguments);

}

class ChangeIncomePageState extends State<ChangeIncomePage> {

var arguments;

ChangeIncomePageState({this.arguments}); //上一页面传值

Bill bill;

//初始化按钮

@override

void initState() {

super.initState();

color[arguments['billTypesId']] = true;

}

var color = [false, false, false]; //判断按钮是否被选中

//改变按钮颜色

void \_set(int index) {

var value = color.indexWhere((element) => element == true);

if (value != -1) {

color[value] = false; //恢复之前按钮颜色

}

color[index] = true; //改变选中按钮颜色

}

//按钮布局及事件

flatButton(int index) {

return FlatButton(

color: Colors.white,

highlightColor: Colors.white,

splashColor: Colors.white,

onPressed: () {

setState(() {

\_set(index); //改变按钮颜色

});

//显示数字键盘

showModalBottomSheet(

context: context,

builder: (context) {

return Container(

child: NumberKeyboard(

bill: Bill(

0,

index,

arguments['billId'],

arguments['billAmount'],

arguments['billYear'],

arguments['billMonth'],

arguments['billDay'],

arguments['billHour'],

arguments['billRemark'],

),

action: 1));

});

},

child: Column(

children: <Widget>[

Container(

color: null,

child: CircleAvatar(

backgroundColor: color[index]

? Color.fromARGB(255, 255, 200, 0)

: Color.fromARGB(255, 239, 239, 239), //按钮颜色

radius: 30.0,

child: Icon(incomeIcons[index], color: Colors.black38), //按钮图标

),

),

Text(incomeTypes[index]) //按钮label

],

),

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

iconTheme: IconThemeData(

color: Colors.black,

), //修改头部图标颜色

title: Text(

"收入",

style: TextStyle(color: Colors.black),

),

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

),

body: Container(

color: Colors.white,

child: GridView.count(

crossAxisSpacing: 8,

mainAxisSpacing: 10,

crossAxisCount: 3,

padding: EdgeInsets.all(20),

children: [flatButton(0), flatButton(1), flatButton(2)], //增加按钮

),

));

}

}

## 2.6 changePay

import 'package:flutter/material.dart';

import 'package:zhizhang/utils/bill.dart';

import 'package:zhizhang/utils/numberKeyboard.dart';

import 'package:zhizhang/utils/types.dart';

// ignore: must\_be\_immutable

class ChangePayPage extends StatefulWidget {

var arguments;

ChangePayPage({Key key, this.arguments}) : super(key: key);

@override

ChangePayPageState createState() =>

ChangePayPageState(arguments: this.arguments);

}

class ChangePayPageState extends State<ChangePayPage> {

var arguments;

ChangePayPageState({this.arguments}); //上一页面传值

Bill bill;

//初始化按钮

@override

void initState() {

super.initState();

color[arguments['billTypesId']] = true;

}

var color = [

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false

]; //判断按钮是否被选中

//数字键盘

//改变按钮颜色

void \_set(int index) {

var value = color.indexWhere((element) => element == true);

if (value != -1) {

color[value] = false; //恢复之前按钮颜色

}

color[index] = true; //改变选中按钮颜色

}

//按钮布局及事件

flatButton(int index) {

return FlatButton(

color: Colors.white,

highlightColor: Colors.white,

splashColor: Colors.white,

onPressed: () {

setState(() {

\_set(index); //改变按钮颜色

});

//显示数字键盘

showModalBottomSheet(

context: context,

builder: (context) {

return Container(

child: NumberKeyboard(

bill: Bill(

1,

index,

arguments['billId'],

arguments['billAmount'],

arguments['billYear'],

arguments['billMonth'],

arguments['billDay'],

arguments['billHour'],

arguments['billRemark'],

),

action: 1),

);

});

},

child: Column(

children: <Widget>[

Container(

color: null,

child: CircleAvatar(

backgroundColor: color[index]

? Color.fromARGB(255, 255, 200, 0)

: Color.fromARGB(255, 239, 239, 239), //按钮颜色

radius: 30.0,

child: Icon(payIcons[index], color: Colors.black38), //按钮图标

),

),

Text(payTypes[index]) //按钮label

],

),

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

iconTheme: IconThemeData(

color: Colors.black,

), //修改头部图标颜色

title: Text(

"支出",

style: TextStyle(color: Colors.black),

),

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

),

body: Container(

color: Colors.white,

//页面按钮布局

child: GridView.count(

crossAxisSpacing: 8,

mainAxisSpacing: 10,

crossAxisCount: 3,

padding: EdgeInsets.all(20),

children: <Widget>[

flatButton(0),

flatButton(1),

flatButton(2),

flatButton(3),

flatButton(4),

flatButton(5),

flatButton(6),

flatButton(7),

flatButton(8),

flatButton(9),

flatButton(10),

flatButton(11),

flatButton(12),

flatButton(13),

flatButton(14),

flatButton(15),

flatButton(16),

flatButton(17),

flatButton(18),

], //增加按钮

)));

}

}

## 2.7 chartIncome

import 'package:flutter/material.dart';

import 'chartIncomeMonth.dart';

import 'chartIncomeYear.dart';

class ChartIncomePage extends StatefulWidget {

@override

ChartIncomePageState createState() => ChartIncomePageState();

}

class ChartIncomePageState extends State<ChartIncomePage> {

@override

Widget build(BuildContext context) {

return DefaultTabController(

length: 2,

child: Scaffold(

appBar: AppBar(

iconTheme: IconThemeData(

color: Colors.black,

), //修改图标颜色

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

elevation: 1, //阴影

title: Row(

children: <Widget>[

Expanded(

child: TabBar(

labelColor: Colors.black,

indicatorColor: Colors.black,

indicatorSize: TabBarIndicatorSize.tab,

tabs: [Tab(text: "月"), Tab(text: "年")])) //头部导航

],

),

),

body: TabBarView(

children: [ChartIncomeMonthPage(), ChartIncomeYearPage()], //显示页面

),

));

}

}

## 2.8 chartIncomeMouth

import 'package:date\_format/date\_format.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_cupertino\_date\_picker/flutter\_cupertino\_date\_picker.dart';

import 'package:charts\_flutter/flutter.dart' as charts;

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class ChartIncomeMonthPage extends StatefulWidget {

@override

ChartIncomeMonthPageState createState() => ChartIncomeMonthPageState();

}

class ChartIncomeMonthPageState extends State<ChartIncomeMonthPage> {

DateTime \_dateTime = DateTime.now(); //时间

double \_sum = 0.00; //总金额

int \_num = 0; //类型数量

var \_daySum = List<double>(); //每日总金额

var \_typeSum = List<double>(); //每种类别总金额

double \_height = 80; //柱形图高度

//每日收入

\_getDay() async {

//改变时间后对数据重置

double \_price;

int \_day;

\_sum = 0.00;

\_daySum.clear();

for (var i = 0; i < 32; i++) {

\_daySum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billDay from bill ' +

'where billYear=${\_dateTime.year} And billMonth=${\_dateTime.month} ' +

'And billFromId=0 and userId=$userId group by billDay'))

.deStream();

results.forEach((element) {

//读取日期及其总金额，并存入\_daySum

setState(() {

\_price = element.byName('sum');

\_sum += \_price;

\_day = element.byName('billDay');

\_daySum[\_day] = \_price;

});

});

conn.close();

}

//每类收入

\_getTypes() async {

//改变时间后对数据重置

double \_price;

int \_type;

\_num = 0;

\_typeSum.clear();

for (var i = 0; i < 3; i++) {

\_typeSum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'Select sum(billAmount) as sum,billTypesId from bill ' +

'where billYear=${\_dateTime.year} And billMonth=${\_dateTime.month} ' +

'And billFromId=0 and userId =$userId group by billTypesId'))

.deStream();

results.forEach((element) {

//读取类别及其总金额，并存入\_typesSum

setState(() {

\_type = element.byName('billTypesId');

\_price = element.byName('sum');

\_typeSum[\_type] = \_price;

\_num++;

\_height = \_num \* 90.0; //根据类型数量设置柱形图高度

});

});

conn.close();

}

//页面内容

\_body() {

List<Widget> list = new List();

list.add(Text(' 总收入：$\_sum')); //总金额

list.add(SizedBox(

height: 20,

)); //中间间隔

list.add(

Column(children: [Container(height: 150, child: \_simpleLine())])); //折线图

list.add(SizedBox(

height: 30,

)); //中间间隔

list.add(Container(

height: 0.5,

width: MediaQuery.of(context).size.width,

color: Colors.grey,

)); //分隔线

list.add(SizedBox(

height: 15,

)); //中间间隔

list.add(Text(

' 收入排行榜',

style: TextStyle(fontSize: 15, fontWeight: FontWeight.w700),

));

//如果有数据，显示柱形图；没有提示没有数据

if (\_num == 0) {

list.add(Container(

child: Column(

children: <Widget>[

SizedBox(

height: 100,

),

Icon(

Icons.assignment,

size: 50,

),

Text("没有数据")

],

),

));

} else {

list.add(Column(

children: [Container(height: \_height, child: \_simpleBar())])); //柱形图

}

list.add(SizedBox(

height: 50,

)); //间隔

return list;

}

//初始化

@override

initState() {

super.initState();

\_getDay();

\_getTypes();

}

//日期选择

\_showDatePicker() {

DatePicker.showDatePicker(

context,

onMonthChangeStartWithFirstDate: false,

pickerTheme: DateTimePickerTheme(

showTitle: true,

confirm: Text('确定',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

cancel: Text('取消',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

),

minDateTime: DateTime.parse('1900-01-01'),

maxDateTime: DateTime.parse('2100-12-31'),

initialDateTime: \_dateTime,

dateFormat: 'yyyy-MMMM',

locale: DateTimePickerLocale.zh\_cn,

onConfirm: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

\_getDay(); //更新时间后刷新

\_getTypes(); //更新时间后刷新

});

},

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Colors.white,

elevation: 1,

title: SizedBox(

width: MediaQuery.of(context).size.width,

child: FlatButton(

child: Text("${formatDate(\_dateTime, [yyyy, ' 年 ', mm, ' 月'])}"),

onPressed: () {

\_showDatePicker();

}),

),

),

body: ListView(

children: \_body(),

),

);

}

Widget \_simpleLine() {

var line = List<TimeSeriesSales>();

//根据月份增加天数

if (\_dateTime.month == 1 ||

\_dateTime.month == 3 ||

\_dateTime.month == 5 ||

\_dateTime.month == 7 ||

\_dateTime.month == 8 ||

\_dateTime.month == 10 ||

\_dateTime.month == 12) {

for (var i = 1; i <= 31; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_daySum[i]));

}

} else if (\_dateTime.month == 4 ||

\_dateTime.month == 6 ||

\_dateTime.month == 9 ||

\_dateTime.month == 11) {

for (var i = 1; i <= 30; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_daySum[i]));

}

} else {

for (var i = 1; i <= 28; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_daySum[i]));

}

}

//折线图数据

var seriesList = [

new charts.Series<TimeSeriesSales, DateTime>(

id: 'line',

colorFn: (\_, \_\_) => charts.ColorUtil.fromDartColor(Colors.black), //颜色

domainFn: (TimeSeriesSales sales, \_) => sales.time,

measureFn: (TimeSeriesSales sales, \_) => sales.sales,

data: line,

)

];

return charts.TimeSeriesChart(seriesList,

animate: true,

defaultRenderer: charts.LineRendererConfig(

// 圆点大小

radiusPx: 3.0,

stacked: false,

// 线的宽度

strokeWidthPx: 1.0,

// 是否显示线

includeLine: true,

// 是否显示圆点

includePoints: true,

// 是否显示包含区域

includeArea: false));

}

//柱状图

Widget \_simpleBar() {

var bar = List<OrdinalSales>();

//增加类型

for (var i = 0; i < 3; i++) {

if (\_typeSum[i] != 0) {

bar.add(new OrdinalSales(incomeTypes[i], \_typeSum[i]));

}

}

//柱形图数据

var seriesList = [

charts.Series<OrdinalSales, String>(

id: 'bar',

colorFn: (\_, \_\_) => charts.ColorUtil.fromDartColor(

Color.fromARGB(255, 255, 200, 0)), //颜色

domainFn: (OrdinalSales sales, \_) => sales.year,

measureFn: (OrdinalSales sales, \_) => sales.sales,

data: bar,

labelAccessorFn: (OrdinalSales sales, \_) => '${sales.sales.toString()}',

insideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.black.darker);

},

outsideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.gray.shadeDefault.darker);

},

)

];

return charts.BarChart(

seriesList,

animate: false,

barGroupingType: charts.BarGroupingType.stacked,

barRendererDecorator: charts.BarLabelDecorator<String>(),

vertical: false, //横向

);

}

}

class TimeSeriesSales {

final DateTime time; //时间

final double sales; //金额

TimeSeriesSales(this.time, this.sales);

}

class OrdinalSales {

final String year; //时间

final double sales; //金额

OrdinalSales(this.year, this.sales);

}

## 2.9 chartIncomeYear

import 'package:date\_format/date\_format.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_cupertino\_date\_picker/flutter\_cupertino\_date\_picker.dart';

import 'package:charts\_flutter/flutter.dart' as charts;

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class ChartIncomeYearPage extends StatefulWidget {

@override

ChartIncomeYearPageState createState() => ChartIncomeYearPageState();

}

class ChartIncomeYearPageState extends State<ChartIncomeYearPage> {

DateTime \_dateTime = DateTime.now(); //时间

double \_sum = 0.00; //总金额

int \_num = 0; //类型数量

var \_monthSum = List<double>(); //每月总金额

var \_typeSum = List<double>(); //每种类别总金额

double \_height = 80; //柱形图高度

//每日收入

\_getDay() async {

//改变时间后对数据重置

double \_price;

int \_month;

\_sum = 0.00;

\_monthSum.clear();

for (var i = 0; i < 13; i++) {

\_monthSum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billMonth from bill ' +

'where billYear=${\_dateTime.year} ' +

'And billFromId=0 and userId=$userId group by billMonth'))

.deStream();

results.forEach((element) {

//读取日期及其总金额，并存入\_monthSum

setState(() {

\_price = element.byName('sum');

\_sum += \_price;

\_month = element.byName('billMonth');

\_monthSum[\_month] = \_price;

});

});

conn.close();

}

//每类收入

\_getTypes() async {

//改变时间后对数据重置

double \_price;

int \_type;

\_num = 0;

\_typeSum.clear();

for (var i = 0; i < 3; i++) {

\_typeSum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billTypesId from bill ' +

'where billYear=${\_dateTime.year} ' +

'And billFromId=0 and userId =$userId group by billTypesId'))

.deStream();

results.forEach((element) {

//读取类别及其总金额，并存入\_typesSum

setState(() {

\_type = element.byName('billTypesId');

\_price = element.byName('sum');

\_typeSum[\_type] = \_price;

\_num++;

\_height = \_num \* 90.0; //根据类型数量设置柱形图高度

});

});

conn.close();

}

//页面内容

\_body() {

List<Widget> list = new List();

list.add(Text(' 总收入：$\_sum')); //总收入

list.add(SizedBox(

height: 20,

)); //中间间隔

list.add(Column(children: [Container(height: 150, child: \_simpleLine())])); //折线图

list.add(SizedBox(

height: 30,

)); //中间间隔

list.add(Container(

height: 0.5,

width: MediaQuery.of(context).size.width,

color: Colors.grey,

)); //分隔线

list.add(SizedBox(

height: 15,

)); //中间间隔

list.add(Text(

' 收入排行榜',

style: TextStyle(fontSize: 15, fontWeight: FontWeight.w700),

));

//如果有数据，显示柱形图；没有提示没有数据

if (\_num == 0) {

list.add(Container(

child: Column(

children: <Widget>[

SizedBox(

height: 100,

),

Icon(

Icons.assignment,

size: 50,

),

Text("没有数据")

],

),

));

} else {

list.add(

Column(children: [Container(height: \_height, child: \_simpleBar())])); //柱形图

}

list.add(SizedBox(

height: 50,

)); //间隔

return list;

}

//初始化

@override

initState() {

super.initState();

\_getDay();

\_getTypes();

}

\_showDatePicker() {

DatePicker.showDatePicker(

context,

onMonthChangeStartWithFirstDate: false,

pickerTheme: DateTimePickerTheme(

showTitle: true,

confirm: Text('确定',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

cancel: Text('取消',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

),

minDateTime: DateTime.parse('1900-01-01'),

maxDateTime: DateTime.parse('2100-12-31'),

initialDateTime: \_dateTime,

dateFormat: 'yyyy',

locale: DateTimePickerLocale.zh\_cn,

onConfirm: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

\_getDay(); //更新时间后刷新

\_getTypes(); //更新时间后刷新

});

},

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Colors.white,

elevation: 1,

title: SizedBox(

width: MediaQuery.of(context).size.width,

child: FlatButton(

child: Text("${formatDate(\_dateTime, [yyyy, ' 年 '])}"),

onPressed: () {

\_showDatePicker();

}),

)),

body: ListView(

children: \_body(),

),

);

}

Widget \_simpleLine() {

var line = List<TimeSeriesSales>();

//增加月份

for (var i = 1; i <= 12; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_monthSum[i]));

}

//折线图数据

var seriesList = [

new charts.Series<TimeSeriesSales, DateTime>(

id: 'line',

colorFn: (\_, \_\_) => charts.ColorUtil.fromDartColor(Colors.black),

domainFn: (TimeSeriesSales sales, \_) => sales.time,

measureFn: (TimeSeriesSales sales, \_) => sales.sales,

data: line,

)

];

return charts.TimeSeriesChart(seriesList,

animate: true,

defaultRenderer: charts.LineRendererConfig(

// 圆点大小

radiusPx: 3.0,

stacked: false,

// 线的宽度

strokeWidthPx: 1.0,

// 是否显示线

includeLine: true,

// 是否显示圆点

includePoints: true,

// 是否显示包含区域

includeArea: false));

}

//柱状图

Widget \_simpleBar() {

var bar = List<OrdinalSales>();

//增加类型

for (var i = 0; i < 3; i++) {

if (\_typeSum[i] != 0) {

bar.add(new OrdinalSales(incomeTypes[i], \_typeSum[i]));

}

}

//柱形图数据

var seriesList = [

charts.Series<OrdinalSales, String>(

id: 'bar',

colorFn: (\_, \_\_) =>

charts.ColorUtil.fromDartColor(Color.fromARGB(255, 255, 200, 0)),

domainFn: (OrdinalSales sales, \_) => sales.year,

measureFn: (OrdinalSales sales, \_) => sales.sales,

data: bar,

labelAccessorFn: (OrdinalSales sales, \_) => '${sales.sales.toString()}',

insideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.black.darker);

},

outsideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.white.darker);

},

)

];

return charts.BarChart(

seriesList,

animate: true,

barGroupingType: charts.BarGroupingType.stacked,

barRendererDecorator: charts.BarLabelDecorator<String>(),

vertical: false,

);

}

}

class TimeSeriesSales {

final DateTime time; //时间

final double sales; //金额

TimeSeriesSales(this.time, this.sales);

}

class OrdinalSales {

final String year; //时间

final double sales; //金额

OrdinalSales(this.year, this.sales);

}

## 2.10 chartPay

import 'package:flutter/material.dart';

import 'package:zhizhang/pages/chartPayMonth.dart';

import 'package:zhizhang/pages/chartPayYear.dart';

class ChartPayPage extends StatefulWidget {

@override

ChartPayPageState createState() => ChartPayPageState();

}

class ChartPayPageState extends State<ChartPayPage> {

@override

Widget build(BuildContext context) {

return DefaultTabController(

length: 2,

child: Scaffold(

appBar: AppBar(

iconTheme: IconThemeData(

color: Colors.black,

), //修改图标颜色

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

elevation: 1, //阴影

title: Row(

children: <Widget>[

Expanded(

child: TabBar(

labelColor: Colors.black,

indicatorColor: Colors.black,

indicatorSize: TabBarIndicatorSize.tab,

tabs: [Tab(text: "月"), Tab(text: "年")])) //头部导航

],

),

),

body: TabBarView(

children: [ChartPayMonthPage(), ChartPayYearPage()], //显示页面

),

));

}

}

## 2.11 chartPayMouth

import 'package:date\_format/date\_format.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_cupertino\_date\_picker/flutter\_cupertino\_date\_picker.dart';

import 'package:charts\_flutter/flutter.dart' as charts;

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class ChartPayMonthPage extends StatefulWidget {

@override

ChartPayMonthPageState createState() => ChartPayMonthPageState();

}

class ChartPayMonthPageState extends State<ChartPayMonthPage> {

DateTime \_dateTime = DateTime.now(); //时间

double \_sum = 0.00; //总金额

int \_num = 0; //类型数量

var \_daySum = List<double>(); //每日总金额

var \_typeSum = List<double>(); //每种类别总金额

double \_height = 80; //柱形图高度

//每日收入

\_getDay() async {

//改变时间后对数据重置

double \_price;

int \_day;

\_sum = 0.00;

\_daySum.clear();

for (var i = 0; i < 32; i++) {

\_daySum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billDay from bill ' +

'where billYear=${\_dateTime.year} And billMonth=${\_dateTime.month} ' +

'And billFromId=1 and userId=$userId group by billDay'))

.deStream();

results.forEach((element) {

setState(() {

\_price = element.byName('sum');

\_sum += \_price;

\_day = element.byName('billDay');

\_daySum[\_day] = \_price;

});

});

conn.close();

}

//每类收入

\_getTypes() async {

//改变时间后对数据重置

double \_price;

int \_type;

\_num = 0;

\_typeSum.clear();

for (var i = 0; i < 19; i++) {

\_typeSum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billTypesId from bill ' +

'where billYear=${\_dateTime.year} And billMonth=${\_dateTime.month} ' +

'And billFromId=1 and userId =$userId group by billTypesId'))

.deStream();

results.forEach((element) {

//读取类别及其总金额，并存入\_typesSum

setState(() {

\_type = element.byName('billTypesId');

\_price = element.byName('sum');

\_typeSum[\_type] = \_price;

\_num++;

\_height = \_num \* 60.0;

});

});

conn.close();

}

//页面内容

\_body() {

List<Widget> list = new List();

list.add(Text(' 总支出：$\_sum'));

list.add(SizedBox(

height: 20,

)); //中间间隔

list.add(Column(children: [Container(height: 150, child: \_simpleLine())])); //折线图

list.add(SizedBox(

height: 30,

)); //中间间隔

list.add(Container(

height: 0.5,

width: MediaQuery.of(context).size.width,

color: Colors.grey,

)); //分隔线

list.add(SizedBox(

height: 15,

)); //中间间隔

list.add(Text(

' 支出排行榜',

style: TextStyle(fontSize: 15, fontWeight: FontWeight.w700),

));

//如果有数据，显示柱形图；没有提示没有数据

if (\_num == 0) {

list.add(Container(

child: Column(

children: <Widget>[

SizedBox(

height: 100,

),

Icon(

Icons.assignment,

size: 50,

),

Text("没有数据")

],

),

));

} else {

list.add(

Column(children: [Container(height: \_height, child: \_simpleBar())])); //柱形图

}

list.add(SizedBox(

height: 50,

)); //间隔

return list;

}

//初始化

@override

initState() {

super.initState();

\_getDay();

\_getTypes();

}

//日期选择

\_showDatePicker() {

DatePicker.showDatePicker(

context,

onMonthChangeStartWithFirstDate: false,

pickerTheme: DateTimePickerTheme(

showTitle: true,

confirm: Text('确定',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

cancel: Text('取消',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

),

minDateTime: DateTime.parse('1900-01-01'),

maxDateTime: DateTime.parse('2100-12-31'),

initialDateTime: \_dateTime,

dateFormat: 'yyyy-MMMM',

locale: DateTimePickerLocale.zh\_cn,

onConfirm: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

\_getDay(); //更新时间后刷新

\_getTypes(); //更新时间后刷新

});

},

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Colors.white,

elevation: 1,

title: SizedBox(

width: MediaQuery.of(context).size.width,

child: FlatButton(

child: Text("${formatDate(\_dateTime, [yyyy, ' 年 ', mm, ' 月'])}"),

onPressed: () {

\_showDatePicker();

}),

),

),

body: ListView(

children: \_body(),

),

);

}

Widget \_simpleLine() {

var line = List<TimeSeriesSales>();

//根据月份增加天数

if (\_dateTime.month == 1 ||

\_dateTime.month == 3 ||

\_dateTime.month == 5 ||

\_dateTime.month == 7 ||

\_dateTime.month == 8 ||

\_dateTime.month == 10 ||

\_dateTime.month == 12) {

for (var i = 1; i <= 31; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_daySum[i]));

}

} else if (\_dateTime.month == 4 ||

\_dateTime.month == 6 ||

\_dateTime.month == 9 ||

\_dateTime.month == 11) {

for (var i = 1; i <= 30; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_daySum[i]));

}

} else {

for (var i = 1; i <= 28; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_daySum[i]));

}

}

//折线图数据

var seriesList = [

new charts.Series<TimeSeriesSales, DateTime>(

id: 'line',

colorFn: (\_, \_\_) => charts.ColorUtil.fromDartColor(Colors.black),

domainFn: (TimeSeriesSales sales, \_) => sales.time,

measureFn: (TimeSeriesSales sales, \_) => sales.sales,

data: line,

)

];

return charts.TimeSeriesChart(seriesList,

animate: true,

defaultRenderer: charts.LineRendererConfig(

// 圆点大小

radiusPx: 3.0,

stacked: false,

// 线的宽度

strokeWidthPx: 1.0,

// 是否显示线

includeLine: true,

// 是否显示圆点

includePoints: true,

// 是否显示包含区域

includeArea: false));

}

//柱状图

Widget \_simpleBar() {

var bar = List<OrdinalSales>();

//增加类型

for (var i = 0; i < 19; i++) {

if (\_typeSum[i] != 0) {

bar.add(new OrdinalSales(payTypes[i], \_typeSum[i]));

}

}

//柱形图数据

var seriesList = [

charts.Series<OrdinalSales, String>(

id: 'bar',

colorFn: (\_, \_\_) =>

charts.ColorUtil.fromDartColor(Color.fromARGB(255, 255, 200, 0)),

domainFn: (OrdinalSales sales, \_) => sales.year,

measureFn: (OrdinalSales sales, \_) => sales.sales,

data: bar,

labelAccessorFn: (OrdinalSales sales, \_) => '${sales.sales.toString()}',

insideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.black.darker);

},

outsideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.gray.shadeDefault.darker);

},

)

];

return charts.BarChart(

seriesList,

animate: false,

barGroupingType: charts.BarGroupingType.stacked,

barRendererDecorator: charts.BarLabelDecorator<String>(),

vertical: false,

);

}

}

class TimeSeriesSales {

final DateTime time; //时间

final double sales; //金额

TimeSeriesSales(this.time, this.sales);

}

class OrdinalSales {

final String year; //时间

final double sales; //金额

OrdinalSales(this.year, this.sales);

}

## 2.12 chartPayYear

import 'package:date\_format/date\_format.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_cupertino\_date\_picker/flutter\_cupertino\_date\_picker.dart';

import 'package:charts\_flutter/flutter.dart' as charts;

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class ChartPayYearPage extends StatefulWidget {

@override

ChartPayYearPageState createState() => ChartPayYearPageState();

}

class ChartPayYearPageState extends State<ChartPayYearPage> {

DateTime \_dateTime = DateTime.now(); //时间

double \_sum = 0.00; //总金额

int \_num = 0; //类型数量

var \_monthSum = List<double>(); //每月总金额

var \_typeSum = List<double>(); //每种类别总金额

double \_height = 80; //柱形图高度

//每日收入

\_getDay() async {

//改变时间后对数据重置

double \_price;

int \_month;

\_sum = 0.00;

\_monthSum.clear();

for (var i = 0; i < 13; i++) {

\_monthSum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billMonth from bill ' +

'where billYear=${\_dateTime.year} ' +

'And billFromId=1 and userId=$userId group by billMonth'))

.deStream();

results.forEach((element) {

//读取日期及其总金额，并存入\_monthSum

setState(() {

\_price = element.byName('sum');

\_sum += \_price;

\_month = element.byName('billMonth');

\_monthSum[\_month] = \_price;

});

});

conn.close();

}

//每类收入

\_getTypes() async {

//改变时间后对数据重置

double \_price;

int \_type;

\_num = 0;

\_typeSum.clear();

for (var i = 0; i < 19; i++) {

\_typeSum.add(0.00);

}

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select sum(billAmount) as sum,billTypesId from bill ' +

'where billYear=${\_dateTime.year} ' +

'And billFromId=1 and userId =$userId group by billTypesId'))

.deStream();

results.forEach((element) {

//读取类别及其总金额，并存入\_typesSum

setState(() {

\_type = element.byName('billTypesId');

\_price = element.byName('sum');

\_typeSum[\_type] = \_price;

\_num++;

\_height = \_num \* 60.0; //根据类型数量设置柱形图高度

});

});

conn.close();

}

//页面内容

\_body() {

List<Widget> list = new List();

list.add(Text(' 总支出：$\_sum')); //总支出

list.add(SizedBox(

height: 20,

)); //中间间隔

list.add(

Column(children: [Container(height: 150, child: \_simpleLine())])); //折线图

list.add(SizedBox(

height: 30,

)); //中间间隔

list.add(Container(

height: 0.5,

width: MediaQuery.of(context).size.width,

color: Colors.grey,

)); //分隔线

list.add(SizedBox(

height: 15,

)); //中间间隔

list.add(Text(

' 支出排行榜',

style: TextStyle(fontSize: 15, fontWeight: FontWeight.w700),

));

//如果有数据，显示柱形图；没有提示没有数据

if (\_num == 0) {

list.add(Container(

child: Column(

children: <Widget>[

SizedBox(

height: 100,

),

Icon(

Icons.assignment,

size: 50,

),

Text("没有数据")

],

),

));

} else {

list.add(Column(

children: [Container(height: \_height, child: \_simpleBar())])); //柱形图

}

list.add(SizedBox(

height: 50,

)); //间隔

return list;

}

//初始化

@override

initState() {

super.initState();

\_getDay();

\_getTypes();

}

\_showDatePicker() {

DatePicker.showDatePicker(

context,

onMonthChangeStartWithFirstDate: false,

pickerTheme: DateTimePickerTheme(

showTitle: true,

confirm: Text('确定',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

cancel: Text('取消',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

),

minDateTime: DateTime.parse('1900-01-01'),

maxDateTime: DateTime.parse('2100-12-31'),

initialDateTime: \_dateTime,

dateFormat: 'yyyy',

locale: DateTimePickerLocale.zh\_cn,

onConfirm: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

\_getDay(); //更新时间后刷新

\_getTypes(); //更新时间后刷新

});

},

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Colors.white,

elevation: 1,

title: SizedBox(

width: MediaQuery.of(context).size.width,

child: FlatButton(

child: Text("${formatDate(\_dateTime, [yyyy, ' 年 '])}"),

onPressed: () {

\_showDatePicker();

}),

)),

body: ListView(

children: \_body(),

),

);

}

Widget \_simpleLine() {

var line = List<TimeSeriesSales>();

//增加月份

for (var i = 1; i <= 12; i++) {

line.add(new TimeSeriesSales(

DateTime(\_dateTime.year, \_dateTime.month, i), \_monthSum[i]));

}

//折线图数据

var seriesList = [

new charts.Series<TimeSeriesSales, DateTime>(

id: 'line',

colorFn: (\_, \_\_) => charts.ColorUtil.fromDartColor(Colors.black),

domainFn: (TimeSeriesSales sales, \_) => sales.time,

measureFn: (TimeSeriesSales sales, \_) => sales.sales,

data: line,

)

];

return charts.TimeSeriesChart(seriesList,

animate: true,

defaultRenderer: charts.LineRendererConfig(

// 圆点大小

radiusPx: 3.0,

stacked: false,

// 线的宽度

strokeWidthPx: 1.0,

// 是否显示线

includeLine: true,

// 是否显示圆点

includePoints: true,

// 是否显示包含区域

includeArea: false));

}

//柱状图

Widget \_simpleBar() {

var bar = List<OrdinalSales>();

//增加类型

for (var i = 0; i < 19; i++) {

if (\_typeSum[i] != 0) {

bar.add(new OrdinalSales(payTypes[i], \_typeSum[i]));

}

}

//柱形图数据

var seriesList = [

charts.Series<OrdinalSales, String>(

id: 'bar',

colorFn: (\_, \_\_) =>

charts.ColorUtil.fromDartColor(Color.fromARGB(255, 255, 200, 0)),

domainFn: (OrdinalSales sales, \_) => sales.year,

measureFn: (OrdinalSales sales, \_) => sales.sales,

data: bar,

labelAccessorFn: (OrdinalSales sales, \_) => '${sales.sales.toString()}',

insideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.black.darker);

},

outsideLabelStyleAccessorFn: (OrdinalSales sales, \_) {

return new charts.TextStyleSpec(

color: charts.MaterialPalette.white.darker);

},

)

];

return charts.BarChart(

seriesList,

animate: true,

barGroupingType: charts.BarGroupingType.stacked,

barRendererDecorator: charts.BarLabelDecorator<String>(),

vertical: false,

);

}

}

class TimeSeriesSales {

final DateTime time; //时间

final double sales; //金额

TimeSeriesSales(this.time, this.sales);

}

class OrdinalSales {

final String year; //时间

final double sales; //金额

OrdinalSales(this.year, this.sales);

}

## 2.13 charts

import 'package:flutter/material.dart';

import 'chartIncome.dart';

import 'chartPay.dart';

class ChartsPage extends StatefulWidget {

@override

ChartsPageState createState() => ChartsPageState();

}

class ChartsPageState extends State<ChartsPage> {

List \_pageList = [ChartPayPage(), ChartIncomePage()]; //页面内容

var \_selectValue;

@override

void initState() {

super.initState();

\_selectValue = getListData()[0].value; //初始化页面内容

}

List<DropdownMenuItem> getListData() {

//头部下拉框

List<DropdownMenuItem> items = new List();

DropdownMenuItem dropdownMenuItem1 = new DropdownMenuItem(

child: new Text('支出'),

value: 0,

);

items.add(dropdownMenuItem1);

DropdownMenuItem dropdownMenuItem2 = new DropdownMenuItem(

child: new Text('收入'),

value: 1,

);

items.add(dropdownMenuItem2);

return items;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

elevation: 0,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

title: Center(

child: DropdownButton(

items: getListData(),

onChanged: (value) {

setState(() {

\_selectValue = value;

});

}, //改变下拉框内容

value: \_selectValue,

underline: Container(), //把底部那条线去掉

),

),

),

body: this.\_pageList[this.\_selectValue], //根据下拉框选项改变页面内容

);

}

}

## 2.14 details

import 'package:date\_format/date\_format.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_cupertino\_date\_picker/flutter\_cupertino\_date\_picker.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/bill.dart';

import 'package:zhizhang/utils/types.dart';

class DetailsPage extends StatefulWidget {

@override

DetailsPageState createState() => DetailsPageState();

}

class DetailsPageState extends State<DetailsPage> {

double \_monthPay = 0.00; //月支出

double \_monthIncome = 0.00; //月收入

var bills = List<Bill>(); //账单列表

var days = List<int>(); //日期列表

//加载数据

\_getBills() async {

//改变时间后对数据重置

\_monthIncome = 0;

\_monthPay = 0;

bills.clear();

days.clear();

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn.execute(

'select \* from bill where userId = $userId and billYear = ${\_dateTime.year} and billMonth = ${\_dateTime.month}'))

.deStream();

results.forEach((element) {

//读取符合条件的账单信息，存入bills

setState(() {

int day = element.byName('billDay');

Bill bill =

Bill(element.byName('billFromId'), element.byName('billTypesId'));

bill.billId = element.byName('billId');

bill.billYear = element.byName('billYear');

bill.billMonth = element.byName('billMonth');

bill.billDay = element.byName('billDay');

bill.billHour = element.byName('billHour');

bill.billAmount = element.byName('billAmount');

bill.billRemark = element.byName('billRemark');

bills.add(bill);

//添加有账单的日期

if (days.contains(day) == false) {

days.add(day);

}

//判断账单类型，分别计算支出与收入的总和

if (bill.billFromId == 1) {

\_monthPay += bill.billAmount;

} else if (bill.billFromId == 0) {

\_monthIncome += bill.billAmount;

}

});

});

//天数排序

days.sort((a, b) {

return a - b;

});

conn.close();

}

//初始化

@override

initState() {

super.initState();

\_getBills();

}

//页面内容

\_body() {

List<Widget> list = new List();

List<Widget> daylist = new List();

list.add(

//顶部

Row(

children: <Widget>[

//月份选择

Container(

width: MediaQuery.of(context).size.width / 3.5,

height: 80,

color: Color.fromARGB(255, 255, 200, 0),

child: FlatButton(

child: Row(

children: <Widget>[

Column(

children: <Widget>[

SizedBox(

height: 10,

),

Text(

"${formatDate(\_dateTime, [yyyy, '年'])}",

style: TextStyle(color: Colors.grey),

),

SizedBox(

height: 10,

),

Text(

"${formatDate(\_dateTime, [mm, '月'])}",

style: TextStyle(fontSize: 20),

),

],

),

// SizedBox(

// width: 1,

// ),

Column(

children: <Widget>[

SizedBox(

height: 40,

),

Icon(Icons.arrow\_drop\_down)

],

),

],

),

onPressed: () {

\_showDatePicker();

}),

),

//分割线

Container(

width: 1,

height: 80,

color: Color.fromARGB(255, 255, 200, 0),

child: Center(

child: Container(

height: 50,

color: Colors.black,

),

),

),

//收入合计

Container(

width: MediaQuery.of(context).size.width / 14 \* 5 - 1,

height: 80,

color: Color.fromARGB(255, 255, 200, 0),

child: Row(

children: <Widget>[

Column(

children: <Widget>[

SizedBox(

height: 10,

),

Text(

" 收入",

style: TextStyle(color: Colors.grey),

),

SizedBox(

height: 10,

),

Expanded(child: Text(

" ${this.\_monthIncome}",

style: TextStyle(fontSize: 20),

)),

],

),

],

)),

//支出合计

Container(

width: MediaQuery.of(context).size.width / 14 \* 5,

height: 80,

color: Color.fromARGB(255, 255, 200, 0),

child: Row(

children: <Widget>[

Column(

children: <Widget>[

SizedBox(

height: 10,

),

Text(

"支出",

style: TextStyle(color: Colors.grey),

),

SizedBox(

height: 10,

),

Expanded(child: Text(

" ${this.\_monthPay}",

style: TextStyle(fontSize: 20),

)),

],

),

],

))

],

),

);

//如果有数据，显示账单；没有提示没有数据

if (bills.isEmpty == true) {

list.add(Container(

child: Column(

children: <Widget>[

SizedBox(

height: 200,

),

Icon(

Icons.assignment,

size: 50,

),

Text("没有数据")

],

),

));

} else {

//遍历天数

for (var i = 0; i < days.length; i++) {

var \_dayIncome = 0.00;

var \_dayPay = 0.00;

//遍历账单

for (var j = 0; j < bills.length; j++) {

if (bills[j].billDay == days[i]) {

if (bills[j].billFromId == 1) {

\_dayPay += bills[j].billAmount;

} else {

\_dayIncome += bills[j].billAmount;

}

daylist.add(ListTile(

leading: Icon(bills[j].billFromId == 1

? payIcons[bills[j].billTypesId]

: incomeIcons[bills[j].billTypesId]), //图标

title: Text(bills[j].billFromId == 1

? payTypes[bills[j].billTypesId]

: incomeTypes[bills[j].billTypesId]), //文本

trailing: Text(bills[j].billFromId == 1

? "-${bills[j].billAmount}"

: "${bills[j].billAmount}"), //金额

onTap: () {

Navigator.pushNamed(context, '/bill', arguments: {

"billId": bills[j].billId,

"billFromId": bills[j].billFromId,

"billTypesId": bills[j].billTypesId,

"billAmount": bills[j].billAmount,

"billYear": bills[j].billYear,

"billMonth": bills[j].billMonth,

"billDay": bills[j].billDay,

"billHour": bills[j].billHour,

"billRemark": bills[j].billRemark

}).then((value) => {

if (value == true) {\_getBills()}

}); //跳转至单个账单明细，如果有更改，返回时刷新

},

));

}

}

list.add(Column(

children: <Widget>[

//每日总计

Row(

children: <Widget>[

// SizedBox(

// width: 10,

// ),

Expanded(child: Text("${\_dateTime.month}月${days[i]}日")),

// SizedBox(

// width: MediaQuery.of(context).size.width / 4,

// ),

Expanded(child: Text("收入：$\_dayIncome")),

// SizedBox(

// width: MediaQuery.of(context).size.width / 18,

// ),

Expanded(child: Text("支出：$\_dayPay"))

],

),

Container(

width: MediaQuery.of(context).size.width,

height: 0.5,

color: Colors.grey,

),

],

));

//每日账单显示

for (var j = 0; j < daylist.length; j++) {

list.add(daylist[j]);

}

daylist.clear();

}

}

return list;

}

DateTime \_dateTime = DateTime.now();

//月份选择

\_showDatePicker() {

DatePicker.showDatePicker(

context,

onMonthChangeStartWithFirstDate: false,

pickerTheme: DateTimePickerTheme(

showTitle: true,

confirm: Text('确定',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

cancel: Text('取消',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

),

minDateTime: DateTime.parse('1900-01-01'),

maxDateTime: DateTime.parse('2100-12-31'),

initialDateTime: \_dateTime,

dateFormat: 'yyyy-MMMM',

locale: DateTimePickerLocale.zh\_cn,

onConfirm: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

\_getBills();

});

},

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

title: Text("制账", style: TextStyle(color: Colors.black)),

iconTheme: IconThemeData(

color: Colors.black,

), //修改图标颜色

actions: [

IconButton(

icon: Icon(

Icons.person\_outline,

),

onPressed: () {

Navigator.pushNamed(context, '/person');

} //跳转个人页面

)

],

),

body: ListView(

children: \_body(),

));

}

}

## 2.15 forgetPwd

import 'package:flutter/material.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class ForgetPwdPage extends StatefulWidget {

@override

ForgetPwdPageState createState() => ForgetPwdPageState();

}

class ForgetPwdPageState extends State<ForgetPwdPage> {

var \_userId = TextEditingController(); //用户名

var \_userPhone = TextEditingController(); //手机号

GlobalKey \_globalKey = new GlobalKey<FormState>();

//判断账号与手机号是否匹配

Future<bool> \_verify() async {

var conn = await MySqlConnection.connect(connect);

Results result = await (await conn.execute(

'select userId from user where phone = ${\_userPhone.text} and userId = ${\_userId.text}'))

.deStream();

//匹配返回false，否则返回true

if (result.isEmpty == true) {

conn.close();

return false;

} else {

conn.close();

return true;

}

}

//密码重置

\_reset() async {

var conn = await MySqlConnection.connect(connect);

await conn.prepared(

'Update user set userPassword = 123456 where userId = ?',

[\_userId.text]);

conn.close();

}

Future<bool> \_forgetPwd() async {

//根据ID与手机是否匹配输出

if (await \_verify() == true) {

\_reset();

showDialog(

context: context,

builder: (context) {

//提示重置成功

return AlertDialog(

title: Text("提示"),

content: Text("密码重置为：123456，请返回登录"),

actions: <Widget>[

FlatButton(

child: Text("确认",

style:

TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

onPressed: () {

Navigator.of(context).pushNamedAndRemoveUntil(

"/login", ModalRoute.withName("/login")); //返回登录页面

},

)

],

);

});

return true;

} else {

//提示账号与手机号不匹配

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("提示"),

content: Text("账号与手机号不匹配"),

actions: <Widget>[

FlatButton(

child: Text("确认",

style:

TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

onPressed: () {

Navigator.of(context).pop(true); //关闭对话框

},

)

],

);

});

return false;

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

iconTheme: IconThemeData(

color: Colors.black, //修改颜色

),

title: Text("忘记密码", style: TextStyle(color: Colors.black)),

),

body: Center(

child: Form(

key: \_globalKey,

// ignore: deprecated\_member\_use

autovalidate: true,

child: Column(children: <Widget>[

SizedBox(

height: 20,

), //中间间隔

//账号输入

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

controller: \_userId,

decoration: InputDecoration(

labelText: "账号",

hintText: "输入你的账号",

icon: Icon(Icons.person)),

)),

),

//手机号输入

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

controller: \_userPhone,

decoration: InputDecoration(

labelText: "手机号",

hintText: "输入你的手机号",

icon: Icon(Icons.phone)),

)),

),

Padding(

padding: EdgeInsets.only(top: 50.0),

),

//找回密码按钮

SizedBox(

width: 300,

height: 50.0,

child: RaisedButton(

onPressed: () {

if ((\_globalKey.currentState as FormState).validate()) {

\_forgetPwd();

}

}, //忘记密码操作

child: Text(

"找回密码",

style: TextStyle(color: Colors.white), //字体白色

),

color: Color.fromARGB(255, 255, 200, 0),

),

),

]))));

}

}

## 2.16 login

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:sqljocky5/connection/connection.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/tabs.dart';

import 'package:zhizhang/utils/types.dart';

class LoginPage extends StatefulWidget {

@override

LoginPageState createState() => LoginPageState();

}

class LoginPageState extends State<LoginPage> {

var \_userId = TextEditingController(); //用户名

var \_userPwd = TextEditingController(); //密码

GlobalKey \_globalKey = new GlobalKey<FormState>(); //用于检查输入框是否为空

Future<int> \_verify() async {

var conn = await MySqlConnection.connect(connect);

Results result = await (await conn.execute(

'select userId from user where userId = ${\_userId.text} and userPassword = ${\_userPwd.text}'))

.deStream();

if (result.isEmpty == false) {

userId = await result.map((r) => r.byName('userId')).single;

}

conn.close();

return userId;

}

Future<bool> \_login() async {

//登录成功，跳转至主页面；登录失败提示

if (await \_verify() != 0) {

Navigator.of(context).pushAndRemoveUntil(

new MaterialPageRoute(builder: (context) => new Tabs()),

(route) => route == null);

return true;

} else {

//对话框提示

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("提示"),

content: Text("账号或密码错误，请检查"),

actions: <Widget>[

FlatButton(

child: Text("确认",

style:

TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

onPressed: () {

Navigator.of(context).pop(true); //关闭对话框

},

)

],

);

});

return false;

}

}

// 保存账号密码

void \_saveLoginMsg() async {

SharedPreferences preferences = await SharedPreferences.getInstance();

preferences.setString("name", \_userId.text);

preferences.setString("pwd", \_userPwd.text);

}

// 读取账号密码，并将值直接赋给账号框和密码框

void \_getLoginMsg() async {

SharedPreferences preferences = await SharedPreferences.getInstance();

\_userId.text = preferences.get("name");

\_userPwd.text = preferences.get("pwd");

}

//初始化

@override

void initState() {

super.initState();

\_getLoginMsg();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

title: Text("登录", style: TextStyle(color: Colors.black)),

),

body: Center(

child: Form(

key: \_globalKey,

// ignore: deprecated\_member\_use

autovalidate: true,

child: Column(

children: <Widget>[

SizedBox(

height: 20,

), //中间间隔

//账号输入

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

key: Key("id"),

controller: \_userId,

decoration: InputDecoration(

labelText: "账号",

hintText: "输入你的账号",

icon: Icon(Icons.person)),

validator: (v) {

return v.trim().length > 0 ? null : "账号不能为空"; //判断账号是否输入

},

)),

),

//密码输入

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

key: Key("password"),

controller: \_userPwd,

decoration: InputDecoration(

labelText: "密码",

hintText: "输入你的密码",

icon: Icon(Icons.lock),

),

validator: (v) {

return v.trim().length > 0 ? null : "密码不能为空"; //判断密码是否输入

},

obscureText: true,

),

)),

SizedBox(

width: double.infinity,

height: 50.0,

child: Row(

//注册按钮

children: <Widget>[

SizedBox(

width: MediaQuery.of(context).size.width / 15,

),

FlatButton(

onPressed: () {

Navigator.pushNamed(context, "/register");

},

child: Text(

"立即注册",

style: TextStyle(color: Colors.black),

),

color: Color.fromARGB(0, 255, 255, 255),

),

SizedBox(

width: MediaQuery.of(context).size.width / 2.8,

), //中间间隔

//忘记密码按钮

FlatButton(

onPressed: () {

Navigator.pushNamed(context, "/forgetPwd");

},

child: Text(

"忘记密码？",

style: TextStyle(color: Colors.black), //字体白色

),

color: Color.fromARGB(0, 255, 255, 255),

),

],

),

),

Padding(

padding: EdgeInsets.only(top: 20.0),

),

//登录按钮

SizedBox(

width: 300,

height: 50.0,

child: RaisedButton(

key: Key('increment'),

onPressed: () {

if ((\_globalKey.currentState as FormState).validate()) {

\_login(); //登录

\_saveLoginMsg(); //缓存账号密码

}

},

child: Text(

"登录",

style: TextStyle(color: Colors.white), //字体白色

),

color: Color.fromARGB(255, 255, 200, 0),

),

),

],

),

),

),

);

}

}

## 2.17 modifyPwd

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class ModifyPwdPage extends StatefulWidget {

@override

ModifyPwdPageState createState() => ModifyPwdPageState();

}

class ModifyPwdPageState extends State<ModifyPwdPage> {

var \_userOldPwd = TextEditingController(); //密码

var \_userPwd = TextEditingController(); //手机号

GlobalKey \_globalKey = new GlobalKey<FormState>();

//判断账号与密码是否匹配

Future<bool> \_verify() async {

var conn = await MySqlConnection.connect(connect);

Results result = await (await conn.execute(

'select userId from user where userId = $userId and userPassword = ${\_userOldPwd.text}'))

.deStream();

//匹配返回false，否则返回true

if (result.isEmpty == true) {

conn.close();

return false;

} else {

conn.close();

return true;

}

}

//修改密码

\_change() async {

var conn = await MySqlConnection.connect(connect);

await conn.prepared('Update user set userPassword = ? where userId = ?',

[\_userPwd.text, userId]);

conn.close();

}

Future<bool> \_modify() async {

//根据旧密码是否错误进行相应操作

if (await \_verify() == true) {

\_change(); //修改密码

//对话框提示修改成功

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("提示"),

content: Text("密码修改完成"),

actions: <Widget>[

FlatButton(

child: Text("确认",

style:

TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

onPressed: () async {

SharedPreferences preferences =

await SharedPreferences.getInstance();

preferences.setString("pwd", \_userPwd.text); //修改缓存中密码

Navigator.of(context).pop(true);

Navigator.of(context).pop(true); //返回个人页面

},

)

],

);

});

return true;

} else {

//对话框提示密码错误

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("提示"),

content: Text("密码错误"),

actions: <Widget>[

FlatButton(

child: Text("确认",

style:

TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

onPressed: () {

Navigator.of(context).pop(true); //关闭对话框

},

)

],

);

});

return false;

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

iconTheme: IconThemeData(

color: Colors.black, //修改颜色

),

title: Text("修改密码", style: TextStyle(color: Colors.black)),

),

body: Center(

child: Form(

key: \_globalKey,

// ignore: deprecated\_member\_use

autovalidate: true,

child: Column(

children: <Widget>[

//旧密码输入

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

controller: \_userOldPwd,

decoration: InputDecoration(

labelText: "旧密码",

hintText: "输入你的密码",

icon: Icon(Icons.lock),

),

obscureText: true,

),

)),

//输入新密码

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

controller: \_userPwd,

decoration: InputDecoration(

labelText: "新密码",

hintText: "输入你的密码",

icon: Icon(Icons.lock),

),

validator: (v) {

//新密码不少于6位

if (v.trim().length != 0) {

if (v.trim().length < 17) {

return v.trim().length > 5 ? null : "密码不低于6位";

} else {

return "密码不大于16位";

}

}

return null;

},

obscureText: true,

),

)),

//确认新密码

Container(

width: 400,

child: Theme(

data: Theme.of(context).copyWith(

primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

decoration: InputDecoration(

labelText: "确认密码",

hintText: "确认你的密码",

icon: Icon(Icons.lock),

),

validator: (v) {

//确认密码

if (v.trim().length != 0) {

return v.trim().compareTo(this.\_userPwd.text) == 0

? null

: "两次密码不同";

}

return null;

},

obscureText: true,

),

)),

Padding(

padding: EdgeInsets.only(top: 50.0),

),

//修改密码按钮

SizedBox(

width: 300,

height: 50.0,

child: RaisedButton(

onPressed: () {

if ((\_globalKey.currentState as FormState).validate()) {

\_modify(); //修改密码

}

},

child: Text(

"修改密码",

style: TextStyle(color: Colors.white), //字体白色

),

color: Color.fromARGB(255, 255, 200, 0),

),

),

],

),

)));

}

}

## 2.18 person

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class PersonPage extends StatefulWidget {

@override

\_PersonPageState createState() => \_PersonPageState();

}

class \_PersonPageState extends State<PersonPage> {

var \_userName = TextEditingController();

var \_userPhone = TextEditingController();

var \_medal;

var \_num;

var change;

GlobalKey \_globalKey = new GlobalKey<FormState>();

//加载个人信息

\_load() async {

var conn = await MySqlConnection.connect(connect);

Results result = await (await conn.execute(

'select userName,phone,num from user where userId = $userId'))

.deStream();

setState(() {

\_userName.text = result.map((r) => r.byName('userName')).single;

\_userPhone.text = result.map((r) => r.byName('phone')).single;

\_num = result.map((r) => r.byName('num')).single;

if (\_num < 10) {

\_medal = '初进记账';

} else if (\_num < 100) {

\_medal = '停不下来';

} else if (\_num < 200) {

\_medal = '记账能手';

} else if (\_num < 300) {

\_medal = '记账小将';

} else if (\_num < 600) {

\_medal = '记账高手';

} else if (\_num < 1000) {

\_medal = '记账大师';

} else if (\_num < 2000) {

\_medal = '记账至尊';

} else {

\_medal = '高处不胜寒';

}

});

conn.close();

}

//修改手机号

\_changePhone() async {

var conn = await MySqlConnection.connect(connect);

await conn.prepared(

'Update user set phone = ? where userId = $userId', [\_userPhone.text]);

conn.close();

}

//修改昵称

\_changeName() async {

var conn = await MySqlConnection.connect(connect);

await conn.prepared('Update user set userName = ? where userId = $userId',

[\_userName.text]);

conn.close();

}

//初始化个人信息

@override

initState() {

super.initState();

\_load();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

iconTheme: IconThemeData(

color: Colors.black, //修改颜色

),

backgroundColor: Color.fromARGB(255, 255, 200, 0),

title: Text(

"个人页面",

style: TextStyle(color: Colors.black),

),

),

body: Center(

child: Column(

children: <Widget>[

SizedBox(

height: 10,

), //间隔

//用户ID

Stack(

alignment: Alignment(0.8, 0),

children: <Widget>[

Container(

color: Colors.white,

child: ListTile(

title: Text('ID'),

),

),

Text(userId.toString(),

style: TextStyle(fontSize: 16, color: Colors.grey))

],

),

Container(

height: 0.5,

width: double.infinity,

color: Colors.grey,

), //间隔

//昵称

Stack(

alignment: Alignment(0.7, 0),

children: <Widget>[

Container(

color: Colors.white,

child: ListTile(

title: Text('昵称'),

trailing: Icon(Icons.keyboard\_arrow\_right),

onTap: () {

//修改昵称

showDialog(

context: context,

builder: (context) {

return Form(

key: \_globalKey,

// ignore: deprecated\_member\_use

autovalidate: true,

child: AlertDialog(

title: Text("修改昵称"),

content: Theme(

data: Theme.of(context).copyWith(

primaryColor:

Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

//controller: \_userName,

decoration: InputDecoration(

labelText: "昵称",

hintText: "输入你的昵称",

), //昵称输入

onChanged: (value) {

setState(() {

change = value;

});

},

validator: (v) {

//昵称不能为空且要小于6位

if (v.trim().length > 0) {

return v.trim().length < 6

? null

: "昵称不能大于6";

}

return "昵称不能为空";

},

)),

actions: <Widget>[

//取消修改

FlatButton(

child: Text("取消",

style: TextStyle(

color: Color.fromARGB(

255, 255, 200, 0))),

onPressed: () {

Navigator.of(context)

.pop(true); //返回个人页面

}),

//确认修改

FlatButton(

child: Text("确认",

style: TextStyle(

color: Color.fromARGB(

255, 255, 200, 0))),

onPressed: () {

if ((\_globalKey.currentState

as FormState)

.validate()) {

setState(() {

\_userName.text = change;

});

\_changeName();

Navigator.of(context)

.pop(true); //返回个人页面

}

},

),

],

));

});

},

),

),

Text("${\_userName.text}",

style: TextStyle(fontSize: 16, color: Colors.grey))

],

),

Container(

height: 0.5,

width: double.infinity,

color: Colors.grey,

), //间隔

Stack(

alignment: Alignment(0.7, 0),

children: <Widget>[

Container(

color: Colors.white,

child: ListTile(

title: Text('手机号'),

trailing: Icon(Icons.keyboard\_arrow\_right),

onTap: () {

showDialog(

context: context,

builder: (context) {

return Form(

key: \_globalKey,

// ignore: deprecated\_member\_use

autovalidate: true,

child: AlertDialog(

title: Text("修改手机号"),

content: Theme(

data: Theme.of(context).copyWith(

primaryColor:

Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

inputFormatters: [

// ignore: deprecated\_member\_use

WhitelistingTextInputFormatter

// ignore: deprecated\_member\_use

.digitsOnly

], //只允许输入数字

decoration: InputDecoration(

labelText: "手机号",

hintText: "输入你的手机号",

),

onChanged: (value) {

setState(() {

change = value;

});

},

validator: (v) {

//手机号是否符合规定

if (v.trim().length != 0) {

return v.trim().length == 11

? null

: "手机号不符合规定";

}

return "手机号不能为空";

},

)),

actions: <Widget>[

//取消修改

FlatButton(

child: Text("取消",

style: TextStyle(

color: Color.fromARGB(

255, 255, 200, 0))),

onPressed: () {

Navigator.of(context)

.pop(true); //返回个人页面

}),

//确认修改

FlatButton(

child: Text("确认",

style: TextStyle(

color: Color.fromARGB(

255, 255, 200, 0))),

onPressed: () {

if ((\_globalKey.currentState

as FormState)

.validate()) {

setState(() {

\_userPhone.text = change;

});

\_changePhone();

Navigator.of(context)

.pop(true); //返回个人页面

}

}),

],

));

});

},

),

),

Text("${\_userPhone.text}",

style: TextStyle(fontSize: 16, color: Colors.grey))

],

), //用户手机号

SizedBox(

height: 50,

), //间隔

Container(

color: Colors.white,

width: double.infinity,

height: 150,

child: Column(

children: [

SizedBox(

height: 10,

),

Row(

children: [

SizedBox(

width: 20,

),

Text(

'记账成就',

style: TextStyle(

fontSize: 16, fontWeight: FontWeight.w600),

),

IconButton(

icon: Icon(

Icons.help\_outline,

size: 20,

),

onPressed: () {

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text('勋章等级'),

content: Container(

child: Table(

columnWidths: const {

0: FixedColumnWidth(20.0),

1: FixedColumnWidth(50.0),

},

border: TableBorder.all(

color: Colors.black,

width: 0.5,

style: BorderStyle.solid,

),

children: [

TableRow(children: [

Text('\t\t\t等级'),

Text('\t\t记账总笔数')

]),

TableRow(children: [

Text('\t\t初进记账'),

Text('\t\t>=0')

]),

TableRow(children: [

Text('\t\t停不下来'),

Text('\t\t>=10')

]),

TableRow(children: [

Text('\t\t记账能手'),

Text('\t\t>=100')

]),

TableRow(children: [

Text('\t\t记账小将'),

Text('\t\t>=200')

]),

TableRow(children: [

Text('\t\t记账高手'),

Text('\t\t>=300')

]),

TableRow(children: [

Text('\t\t记账大师'),

Text('\t\t>=600')

]),

TableRow(children: [

Text('\t\t记账至尊'),

Text('\t\t>=1000')

]),

TableRow(children: [

Text('\t\t高处不胜寒'),

Text('\t\t>=2000')

]),

],

),

),

actions: [

FlatButton(

child: Text("确认",

style: TextStyle(

color: Color.fromARGB(

255, 255, 200, 0))),

onPressed: () {

Navigator.of(context)

.pop(true); //关闭对话框

})

],

);

});

})

],

),

SizedBox(

height: 10,

),

Row(

children: [

Expanded(

child: Column(

children: [

Text(

\_medal.toString(),

style: TextStyle(

fontSize: 16, fontWeight: FontWeight.w600),

),

SizedBox(

height: 10,

),

Text('勋章',

style: TextStyle(

fontSize: 14,

color: Colors.grey,

))

],

),

),

Expanded(

child: Column(

children: [

Text(

'$\_num笔',

style: TextStyle(

fontSize: 16, fontWeight: FontWeight.w600),

),

SizedBox(

height: 10,

),

Text(

'记账总笔数',

style: TextStyle(

fontSize: 14,

color: Colors.grey,

),

)

],

),

)

],

)

],

),

),

SizedBox(

height: 50,

),

//修改密码按钮

Container(

width: double.infinity,

height: 50,

child: RaisedButton(

child: Text('修改密码'),

textColor: Colors.black,

color: Colors.white,

elevation: 10,

onPressed: () {

Navigator.pushNamed(context, "/modify"); //跳转至修改密码页面

},

),

),

//退出登录按钮

Container(

width: double.infinity,

height: 50,

child: RaisedButton(

child: Text('退出登录'),

textColor: Colors.red,

color: Colors.white,

elevation: 10,

onPressed: () {

//对话框提示

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: Text("提示"),

content: Text("退出后不会删除任何数据，下次登录依然可以使用本账号"),

actions: <Widget>[

FlatButton(

child: Text("取消",

style: TextStyle(

color:

Color.fromARGB(255, 255, 200, 0))),

onPressed: () {

Navigator.of(context).pop(true); //返回个人页面

}),

FlatButton(

child: Text("退出登录",

style: TextStyle(

color: Color.fromARGB(255, 255, 200, 0))),

onPressed: () {

userId = 0;

Navigator.of(context).pushNamedAndRemoveUntil(

"/login",

ModalRoute.withName("/login")); //返回登录页面

},

)

],

);

},

);

}),

)

],

)));

}

}

## 2.19 register

**import 'package:flutter/material.dart'**;  
**import 'package:flutter/services.dart'**;  
**import 'package:sqljocky5/sqljocky.dart'**;  
**import 'package:zhizhang/utils/types.dart'**;  
  
**class** RegisterPage **extends** StatefulWidget {  
 @override  
 RegisterPageState createState() => RegisterPageState();  
}  
  
**class** RegisterPageState **extends** State<RegisterPage> {  
 int **\_userId** = 0;  
 **var \_userName** = TextEditingController(); *//用户名* **var \_userPwd** = TextEditingController(); *//密码* **var \_userPhone** = TextEditingController(); *//手机号* GlobalKey **\_globalKey** = **new** GlobalKey<FormState>();  
  
 *//判断手机号是否被注册* Future<bool> \_verify() **async** {  
 **var** conn = **await** MySqlConnection.*connect*(connect);  
 Results result = **await** (**await** conn.execute(  
 **'select userId from user where phone =** ${**\_userPhone**.**text**}**'**))  
 .deStream();  
  
 **if** (result.**isEmpty** == **true**) {  
 conn.close();  
 **return false**;  
 } **else** {  
 conn.close();  
 **return true**;  
 }  
 }  
  
*//注册* \_add() **async** {  
 **var** conn = **await** MySqlConnection.*connect*(connect);  
 Results result =  
 **await** (**await** conn.execute(**'Select max(userId) as max from user'**))  
 .deStream();  
 **\_userId** = result.map((r) => r.byName(**'max'**)).**single** + 1;  
  
 **await** conn.prepared(  
 **'Insert into user (userId,username,userPassword,phone)values(?,?,?,?)'**,  
 [**\_userId**, **\_userName**.**text**, **\_userPwd**.**text**, **\_userPhone**.**text**]);  
 conn.close();  
 }  
  
 *//根据手机号是否被注册进行操作* Future<bool> \_register() **async** {  
 **if** (**await** \_verify() == **false**) {  
 *//手机号未被注册* **await** \_add();  
 showDialog(  
 context: **context**,  
 builder: (context) {  
 **return** AlertDialog(  
 title: Text(**"提示"**),  
 content: Text(**"注册完成，您的ID为**$**\_userId，该ID用于登陆或忘记密码，请注意保存"**),  
 actions: <Widget>[  
 FlatButton(  
 child: Text(**"确认"**,  
 style:  
 TextStyle(color: Color.fromARGB(255, 255, 200, 0))),  
 onPressed: () {  
 Navigator.*of*(context).pushNamedAndRemoveUntil(  
 **"/login"**, ModalRoute.*withName*(**"/login"**)); *//返回登录页面* },  
 )  
 ],  
 );  
 });  
 **return true**;  
 } **else** {  
 *//手机号被注册* showDialog(  
 context: **context**,  
 builder: (context) {  
 **return** AlertDialog(  
 title: Text(**"提示"**),  
 content: Text(**"手机号已注册"**),  
 actions: <Widget>[  
 FlatButton(  
 child: Text(**"确认"**,  
 style:  
 TextStyle(color: Color.fromARGB(255, 255, 200, 0))),  
 onPressed: () {  
 Navigator.*of*(context).pop(**true**); *//关闭对话框* },  
 )  
 ],  
 );  
 });  
 **return false**;  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 appBar: AppBar(  
 centerTitle: **true**,  
 iconTheme: IconThemeData(  
 color: Colors.*black*, *//修改颜色* ),  
 backgroundColor: Color.fromARGB(255, 255, 200, 0),  
 title: Text(**"注册"**, style: TextStyle(color: Colors.*black*)),  
 ),  
 body: Center(  
 child: Form(  
 key: **\_globalKey**,  
 *// ignore: deprecated\_member\_use* autovalidate: **true**,  
 child: Column(  
 children: <Widget>[  
 Container(  
 width: 400,  
 child: Theme(  
 data: Theme.*of*(context)  
 .copyWith(primaryColor: Color.fromARGB(255, 255, 200, 0)),  
 child: TextFormField(  
 controller: **\_userName**,  
 decoration: InputDecoration(  
 labelText: **"昵称"**,  
 hintText: **"输入你的昵称"**,  
 icon: Icon(Icons.*person*)),  
 validator: (v) {  
 *//昵称不能为空且要小于6位* **if** (v.trim().**length** > 0) {  
 **return** v.trim().**length** < 7 ? **null** : **"昵称不能大于6位"**;  
 }  
 **return "昵称不能为空"**;  
 },  
 )), *//输入昵称* ),  
 Container(  
 width: 400,  
 child: Theme(  
 data: Theme.*of*(context)  
 .copyWith(primaryColor: Color.fromARGB(255, 255, 200, 0)),  
 child: TextFormField(  
 controller: **\_userPwd**,  
 decoration: InputDecoration(  
 labelText: **"密码"**,  
 hintText: **"输入你的密码"**,  
 icon: Icon(Icons.*lock*),  
 ),  
 validator: (v) {  
 *//密码不低于6位* **if** (v.trim().**length** != 0) {  
 **if** (v.trim().**length** < 17) {  
 **return** v.trim().**length** > 5 ? **null** : **"密码不低于6位"**;  
 } **else** {  
 **return "密码不大于16位"**;  
 }  
 }  
 **return null**;  
 },  
 obscureText: **true**,  
 ),  
 )), *//输入密码* Container(  
 width: 400,  
 child: Theme(  
 data: Theme.*of*(context)  
 .copyWith(primaryColor: Color.fromARGB(255, 255, 200, 0)),  
 child: TextFormField(  
 decoration: InputDecoration(  
 labelText: **"确认密码"**,  
 hintText: **"确认你的密码"**,  
 icon: Icon(Icons.*lock*),  
 ),  
 validator: (v) {  
 *//两次密码不同* **if** (v.trim().**length** != 0) {  
 **return** v.trim().compareTo(**this**.**\_userPwd**.**text**) == 0  
 ? **null** : **"两次密码不同"**;  
 }  
 **return null**;  
 },  
 obscureText: **true**,  
 ),  
 )), *//确认密码* Container(  
 width: 400,  
 child: Theme(  
 data: Theme.*of*(context)  
 .copyWith(primaryColor: Color.fromARGB(255, 255, 200, 0)),  
 child: TextFormField(  
 inputFormatters: [  
 *// ignore: deprecated\_member\_use* WhitelistingTextInputFormatter.*digitsOnly* ], *//只允许输入数字* controller: **\_userPhone**,  
 decoration: InputDecoration(  
 labelText: **"手机号"**,  
 hintText: **"输入你的手机号"**,  
 icon: Icon(Icons.*phone*)),  
 validator: (v) {  
 *//手机号是否符合规定* **if** (v.trim().**length** != 0) {  
 **return** v.trim().**length** == 11 ? **null** : **"手机号不符合规定"**;  
 }  
 **return null**;  
 },  
 )),  
 ),  
 Padding(  
 padding: EdgeInsets.only(top: 50.0),  
 ),  
  
 *//注册按钮* SizedBox(  
 width: 300,  
 height: 50.0,  
 child: RaisedButton(  
 onPressed: () {  
 **if** ((**\_globalKey**.**currentState as** FormState).validate()) {  
 \_register();  
 }  
 },  
 child: Text(  
 **"注册"**,  
 style: TextStyle(color: Colors.*white*), *//字体白色* ),  
 color: Color.fromARGB(255, 255, 200, 0),  
 ),  
 ),  
 ],  
 ),  
 )),  
 );  
 }  
}

## 2.20 bill

class Bill {

int billId; //账单id

int billTypesId; //账单类型

int billFromId; //账单类别

double billAmount; //账单金额

int billYear; //账单年

int billMonth; //账单月

int billDay; //账单日

int billHour; //账单时

String billRemark; //账单备注

Bill(this.billFromId, this.billTypesId,

[this.billId,

this.billAmount,

this.billYear,

this.billMonth,

this.billDay,

this.billHour,

this.billRemark]);

}

## 2.21 numberKeyboard

import 'package:flutter/material.dart';

import 'package:flutter\_cupertino\_date\_picker/flutter\_cupertino\_date\_picker.dart';

import 'package:date\_format/date\_format.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/tabs.dart';

import 'package:zhizhang/utils/types.dart';

import 'bill.dart';

//账单增加键盘

// ignore: must\_be\_immutable

class NumberKeyboard extends StatefulWidget {

Bill bill;

int action;

NumberKeyboard({

Key key,

@required this.bill,

@required this.action,

}) : super(key: key);

@override

State createState() => new \_NumberKeyboardState();

}

class \_NumberKeyboardState extends State<NumberKeyboard> {

var \_price = TextEditingController();

var \_remark = TextEditingController(); //备注

DateTime \_dateTime; //时间

///键盘上的键值名称

static const List<String> \_keyNames = [

'7',

'8',

'9',

'4',

'5',

'6',

'1',

'2',

'3',

'.',

'0',

'<-'

];

//增加账单

\_add() async {

var \_money = double.parse(\_price.text);

var conn = await MySqlConnection.connect(connect);

await conn.prepared(

'Insert into bill (billTypesId,billFromId,userId,billAmount,billYear,billMonth,billDay,billHour,billRemark) values (?,?,?,?,?,?,?,?,?)',

[

this.widget.bill.billTypesId,

this.widget.bill.billFromId,

userId,

\_money,

\_dateTime.year,

\_dateTime.month,

\_dateTime.day,

\_dateTime.hour,

\_remark.text

]);

conn.close();

}

//修改账单

\_modify() async {

var \_money = double.parse(\_price.text);

var conn = await MySqlConnection.connect(connect);

await conn.prepared(

'Update bill set billTypesId=?, billFromId=?, userId=?, billAmount=?, billYear=?, billMonth=?, billDay=?, billHour=?, billRemark=? where billId =?',

[

this.widget.bill.billTypesId,

this.widget.bill.billFromId,

userId,

\_money,

\_dateTime.year,

\_dateTime.month,

\_dateTime.day,

\_dateTime.hour,

\_remark.text,

this.widget.bill.billId

]);

conn.close();

}

//初始化：增加账单时，金额为0，时间为当前时间；修改账单时，信息为账单信息

@override

void initState() {

super.initState();

if (this.widget.action == 0) {

\_price.text = '0.00';

\_dateTime = DateTime.now();

} else {

\_price.text = this.widget.bill.billAmount.toString();

\_remark.text = this.widget.bill.billRemark;

var hour;

if (this.widget.bill.billHour < 10) {

hour = "0" + "${this.widget.bill.billHour}";

} else {

hour = "${this.widget.bill.billHour}";

}

\_dateTime = DateTime.parse(

'${this.widget.bill.billYear}-${this.widget.bill.billMonth}-${this.widget.bill.billDay} $hour:00:00');

}

}

//时间

\_showDatePicker() {

DatePicker.showDatePicker(

context,

onMonthChangeStartWithFirstDate: false,

pickerTheme: DateTimePickerTheme(

showTitle: true,

confirm: Text('确定',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

cancel: Text('取消',

style: TextStyle(color: Color.fromARGB(255, 255, 200, 0))),

),

minDateTime: DateTime.parse('1900-01-01 00:00:00'), //最小日期

maxDateTime: DateTime.parse('2100-12-31 23:59:59'), //最大日期

initialDateTime: \_dateTime,

dateFormat: 'yyyy-MMMM-dd-HH',

pickerMode: DateTimePickerMode.datetime,

locale: DateTimePickerLocale.zh\_cn,

onChange: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

});

},

onConfirm: (dateTime, List<int> index) {

setState(() {

\_dateTime = dateTime;

});

},

);

}

///控件点击事件

void \_onViewClick(String keyName) {

var currentText = \_price.text; //当前的文本

//如果第一位是数字0，那么第二次输入的是1-9，那么就替换

if (currentText == '0.00' && (RegExp('^[1-9]\$').hasMatch(keyName))) {

\_price.text = keyName;

return;

}

//如果第一位是数字0，那么第二次输入的是1-9，那么就替换

if (currentText == '0' && (RegExp('^[1-9]\$').hasMatch(keyName))) {

\_price.text = keyName;

return;

}

//如果第一位是数字0，输入的是0，不做反应

if (currentText == '0' && (RegExp('^[0]\$').hasMatch(keyName))) {

\_price.text = keyName;

return;

}

//已有小数点，且小数点后有两位，不做反应

if (RegExp('^\\d+\\.\\d{2}\$').hasMatch(currentText) && keyName != '<-') {

return;

}

//不能第一个就输入.或者<-,不能在已经输入了.再输入

if ((currentText == '' && (keyName == '.' || keyName == '<-')) ||

(RegExp('\\.').hasMatch(currentText) && keyName == '.')) {

return;

}

//撤销

if (keyName == '<-') {

//值为0.00，不做反应

if (currentText == '0.00') {

return;

}

//值如果只有一位，返回1；否则删除最后一位

if (currentText.length == 1) {

\_price.text = '0';

} else {

\_price.text = currentText.substring(0, currentText.length - 1);

}

return;

}

//金额不得超过7位

if (currentText.length > 6) {

return;

}

\_price.text = currentText + keyName;

}

///数字展示面板

Widget \_showDigitalView() {

return Container(

height: 80.0,

padding: const EdgeInsets.all(10.0),

color: Colors.white,

child: Row(

children: <Widget>[

//输入备注

Container(

width: MediaQuery.of(context).size.width / 4.0 \* 3,

child: Theme(

data: Theme.of(context)

.copyWith(primaryColor: Color.fromARGB(255, 255, 200, 0)),

child: TextFormField(

controller: \_remark,

decoration: InputDecoration(

labelText: "备注",

hintText: "点击写备注..",

icon: Icon(Icons.bookmark\_outline)),

)),

),

//输入的数字

Expanded(

child: TextField(

enabled: false,

textAlign: TextAlign.center,

controller: \_price,

decoration: InputDecoration(

border: InputBorder.none,

hintStyle: TextStyle(

color: Color(0xeaeaeaea),

fontSize: 18,

letterSpacing: 2.0),

contentPadding: const EdgeInsets.all(0.0)),

),

),

],

),

);

}

///构建显示数字键盘的视图

Widget \_showKeyboardGridView() {

List<Widget> keyWidgets = new List();

for (int i = 0; i < \_keyNames.length; i++) {

keyWidgets.add(

Material(

color: Colors.white,

child: InkWell(

onTap: () => \_onViewClick(\_keyNames[i]),

child: Container(

width: MediaQuery.of(context).size.width / 4.0,

height: 50,

child: Center(

child: i == \_keyNames.length - 1

? Icon(Icons.backspace)

: Text(

\_keyNames[i],

style: TextStyle(

fontSize: 16,

color: Color(

0xff606060,

),

),

),

),

),

),

),

);

}

return Wrap(children: keyWidgets);

}

@override

Widget build(BuildContext context) {

return IntrinsicHeight(

child: Container(

height: 300,

width: MediaQuery.of(context).size.width,

child: Column(

children: <Widget>[

\_showDigitalView(), //备注与金额显示

Row(

children: <Widget>[

//数字键盘

Container(

width: MediaQuery.of(context).size.width / 4.0 \* 3,

height: 200,

child: \_showKeyboardGridView(),

),

Column(

children: <Widget>[

//时间选择

Container(

width: MediaQuery.of(context).size.width / 4.0,

height: 50,

child: FlatButton(

child: Text("${formatDate(\_dateTime, [

yyyy,

'-',

mm,

'-',

dd,

'\n\t\t\t\t\t',

HH,

'时'

])}"),

onPressed: () {

\_showDatePicker();

}),

),

//完成按钮

Container(

width: MediaQuery.of(context).size.width / 4.0,

height: 150,

color: Color.fromARGB(255, 255, 200, 0),

child: FlatButton(

child: Text("完 成", style: TextStyle(fontSize: 15)),

onPressed: () async {

if (\_price.text != '0.00') {

if (this.widget.action == 0) {

await \_add();

Navigator.of(context).pushAndRemoveUntil(

new MaterialPageRoute(

builder: (context) => new Tabs()),

(route) => route == null); //返回明细页面

} else if (this.widget.action == 1) {

await \_modify();

Navigator.of(context).pop();

Navigator.of(context).pop(true); //返回账单详情

}

}

}),

),

],

)

],

)

],

),

),

);

}

}

## 2.22 routes

import 'package:flutter/material.dart';

import 'package:zhizhang/pages/login.dart';

import 'package:zhizhang/pages/add.dart';

import 'package:zhizhang/pages/modifyPWD.dart';

import 'package:zhizhang/pages/person.dart';

import 'package:zhizhang/pages/register.dart';

import 'package:zhizhang/pages/forgetPwd.dart';

import 'package:zhizhang/pages/billDetail.dart';

import 'package:zhizhang/pages/changeIncome.dart';

import 'package:zhizhang/pages/changePay.dart';

import 'tabs.dart';

//路由

final routes = {

//"/bill": (context, {arguments}) => BillDatilPage(arguments: arguments), //账单明细

//"/changeIncome": (context, {arguments}) =>

// ChangeIncomePage(arguments: arguments), //修改收入页面

"/changePay": (context, {arguments}) =>

ChangePayPage(arguments: arguments), //修改支出页面

"/modify": (context) => ModifyPwdPage(), //修改密码页面

"/login": (context) => LoginPage(), //登录页面

"/add": (context) => AddPage(), //增加账单页面

"/person": (context) => PersonPage(), //个人页面

"/register": (context) => RegisterPage(), //注册页面

"/forgetPwd": (context) => ForgetPwdPage(), //忘记密码页面

"/": (context) => Tabs() //主页面

};

var onGenerateRoute = (RouteSettings settings) {

// 统一处理

final String name = settings.name;

final Function pageContentBuilder = routes[name];

if (pageContentBuilder != null) {

if (settings.arguments != null) {

final Route route = MaterialPageRoute(

builder: (context) =>

pageContentBuilder(context, arguments: settings.arguments));

return route;

} else {

final Route route =

MaterialPageRoute(builder: (context) => pageContentBuilder(context));

return route;

}

}

};

## 2.23 tabs

import 'package:flutter/material.dart';

import 'package:zhizhang/pages/details.dart';

import 'package:zhizhang/pages/charts.dart';

import 'package:zhizhang/pages/add.dart';

class Tabs extends StatefulWidget {

@override

\_TabsState createState() => \_TabsState();

}

class \_TabsState extends State<Tabs> {

int \_currentIndex = 0; //页面值

List \_pageList = [DetailsPage(), AddPage(), ChartsPage()]; //页面内容

@override

Widget build(BuildContext context) {

return Scaffold(

//浮动增加按钮

floatingActionButton: Container(

padding: EdgeInsets.all(5),

decoration: BoxDecoration(

borderRadius: BorderRadius.circular(40), color: Colors.white),

child: FloatingActionButton(

child: Icon(Icons.add),

backgroundColor: Color.fromARGB(255, 255, 200, 0),

onPressed: () {

Navigator.pushNamed(context, "/add");

},

),

),

floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

body: this.\_pageList[this.\_currentIndex], //内容页面

bottomNavigationBar: BottomNavigationBar(

currentIndex: this.\_currentIndex,

onTap: (int index) {

//改变页面内容

if (index == 1) {

Navigator.pushNamed(context, "/add");

} else {

setState(() {

this.\_currentIndex = index;

});

}

},

fixedColor: Color.fromARGB(255, 255, 200, 0),

items: [

BottomNavigationBarItem(

icon: Icon(Icons.assignment),

// ignore: deprecated\_member\_use

title: Text("明细")),

BottomNavigationBarItem(

icon: Icon(Icons.add),

// ignore: deprecated\_member\_use

title: Text("添加")),

BottomNavigationBarItem(

icon: Icon(Icons.bar\_chart\_outlined),

// ignore: deprecated\_member\_use

title: Text("图表")),

], //底部导航条内容

),

);

}

}

## 2.24 types

import 'package:flutter/material.dart';

import 'package:sqljocky5/sqljocky.dart';

int userId = 0; //用户id

//数据库连接

var connect = ConnectionSettings(

user: "root",

password: "y652x676R",

host: "mlcmagic.mysql.rds.aliyuncs.com",

port: 3306,

db: "zhizhang",

);

//收入图标

var incomeIcons = [

IconData(0xe615, fontFamily: 'MyIcons'),

IconData(0xe697, fontFamily: 'MyIcons'),

IconData(0xe629, fontFamily: 'MyIcons')

];

//收入类型

var incomeTypes = ["工资", "理财", "礼金"];

//支出图标

var payIcons = [

IconData(0xe67b, fontFamily: 'MyIcons'),

IconData(0xe646, fontFamily: 'MyIcons'),

IconData(0xe60e, fontFamily: 'MyIcons'),

IconData(0xe618, fontFamily: 'MyIcons'),

IconData(0xe607, fontFamily: 'MyIcons'),

IconData(0xe616, fontFamily: 'MyIcons'),

IconData(0xe61c, fontFamily: 'MyIcons'),

IconData(0xe679, fontFamily: 'MyIcons'),

IconData(0xe63b, fontFamily: 'MyIcons'),

IconData(0xe61d, fontFamily: 'MyIcons'),

IconData(0xe85f, fontFamily: 'MyIcons'),

IconData(0xe609, fontFamily: 'MyIcons'),

IconData(0xe506, fontFamily: 'MyIcons'),

IconData(0xe69b, fontFamily: 'MyIcons'),

IconData(0xe743, fontFamily: 'MyIcons'),

IconData(0xe70e, fontFamily: 'MyIcons'),

IconData(0xe611, fontFamily: 'MyIcons'),

IconData(0xe606, fontFamily: 'MyIcons'),

IconData(0xe677, fontFamily: 'MyIcons')

];

//支出类型

var payTypes = [

"餐饮美食",

"服饰美容",

"生活用品",

"充值缴费",

"交通出行",

"通讯物流",

"公益捐赠",

"医疗健康",

"住房物业",

"图书教育",

"酒店旅行",

"爱车养车",

"运动户外",

"生活服务",

"投资理财",

"保险",

"信用借还",

"人情往来",

"自定义"

];

## 2.25 pubspec

name: zhizhang

description: A new Flutter application.

# The following line prevents the package from being accidentally published to

# pub.dev using `pub publish`. This is preferred for private packages.

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

version: 1.0.0+1

environment:

sdk: ">=2.7.0 <3.0.0"

dependencies:

flutter:

sdk: flutter

# The following adds the Cupertino Icons font to your application.

# Use with the CupertinoIcons class for iOS style icons.

cupertino\_icons: ^1.0.0

shared\_preferences: ^0.5.12+4

flutter\_cupertino\_date\_picker: ^1.0.26+2

date\_format: ^1.0.9

sqljocky5: ^2.2.1

connectivity: ^2.0.2

flutter\_spinkit: ^4.1.2+1

charts\_flutter: ^0.9.0

dev\_dependencies:

flutter\_test:

sdk: flutter

flutter\_driver:

sdk: flutter

test: any

# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.

flutter:

# The following line ensures that the Material Icons font is

# included with your application, so that you can use the icons in

# the material Icons class.

uses-material-design: true

fonts:

- family: MyIcons

fonts:

- asset: assets/logo/iconfont.ttf

assets:

- images/1.jpg

- images/2.0x/1.jpg

- images/3.0x/1.jpg

- images/4.0x/1.jpg

# To add assets to your application, add an assets section, like this:

# :

# - images/a\_dot\_burr.jpeg

# - images/a\_dot\_ham.jpeg

# An image asset can refer to one or more resolution-specific "variants", see

# https://flutter.dev/assets-and-images/#resolution-aware.

# For details regarding adding assets from package dependencies, see

# https://flutter.dev/assets-and-images/#from-packages

# To add custom fonts to your application, add a fonts section here,

# in this "flutter" section. Each entry in this list should have a

# "family" key with the font family name, and a "fonts" key with a

# list giving the asset and other descriptors for the font. For

# example:

# fonts:

# - family: Schyler

# fonts:

# - asset: fonts/Schyler-Regular.ttf

# - asset: fonts/Schyler-Italic.ttf

# style: italic

# - family: Trajan Pro

# fonts:

# - asset: fonts/TrajanPro.ttf

# - asset: fonts/TrajanPro\_Bold.ttf

# weight: 700

#

# For details regarding fonts from package dependencies,

# see <https://flutter.dev/custom-fonts/#from-packages>

## 2.26 rank

import 'package:flutter/material.dart';

import 'package:sqljocky5/sqljocky.dart';

import 'package:zhizhang/utils/types.dart';

class RankPage extends StatefulWidget {

@override

\_RankPageState createState() => \_RankPageState();

}

class \_RankPageState extends State<RankPage> {

var users = List<User>();

User \_use = User(0, '0', -1, '0');

int \_rank = 0;

int \_flag = 0;

\_body() {

List<Widget> list = new List();

if (\_flag == 1) {

if (\_use.rank == 1) {

list.add(ListTile(

leading: Icon(IconData(0xe928, fontFamily: 'MyIcons')),

title: Text(\_use.name),

subtitle: Text(

'${\_use.medal}',

style: TextStyle(fontSize: 12),

),

trailing: Text(

'${\_use.num}',

style: TextStyle(fontSize: 16),

),

));

} else if (\_use.rank == 2) {

list.add(ListTile(

leading: Icon(IconData(0xe929, fontFamily: 'MyIcons')),

title: Text(\_use.name),

subtitle: Text('${\_use.medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${\_use.num}', style: TextStyle(fontSize: 16)),

));

} else if (\_use.rank == 3) {

list.add(ListTile(

leading: Icon(IconData(0xe92a, fontFamily: 'MyIcons')),

title: Text(\_use.name),

subtitle: Text('${\_use.medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${\_use.num}', style: TextStyle(fontSize: 16)),

));

} else {

list.add(ListTile(

leading: Text('${\_use.rank}'),

title: Text(\_use.name),

subtitle: Text('${\_use.medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${\_use.num}', style: TextStyle(fontSize: 16)),

));

}

}

list.add(

Container(

width: MediaQuery.of(context).size.width,

height: 0.2,

color: Colors.grey,

),

);

list.add(SizedBox(

height: 50,

));

list.add(

Container(

width: MediaQuery.of(context).size.width,

height: 0.2,

color: Colors.grey,

),

);

for (var i = 0; i < users.length; i++) {

if (i >= 100) {

break;

}

if (i == 0) {

list.add(ListTile(

leading: Icon(IconData(0xe928, fontFamily: 'MyIcons')),

title: Text(users[i].name),

subtitle: Text('${users[i].medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${users[i].num}', style: TextStyle(fontSize: 16)),

));

} else if (i == 1) {

list.add(ListTile(

leading: Icon(IconData(0xe929, fontFamily: 'MyIcons')),

title: Text(users[i].name),

subtitle: Text('${users[i].medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${users[i].num}', style: TextStyle(fontSize: 16)),

));

} else if (i == 2) {

list.add(ListTile(

leading: Icon(IconData(0xe92a, fontFamily: 'MyIcons')),

title: Text(users[i].name),

subtitle: Text('${users[i].medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${users[i].num}', style: TextStyle(fontSize: 16)),

));

} else {

list.add(ListTile(

leading: Text('\t\t${i + 1}'),

title: Text(users[i].name),

subtitle: Text('${users[i].medal}', style: TextStyle(fontSize: 12)),

trailing: Text('${users[i].num}', style: TextStyle(fontSize: 16)),

));

}

list.add(

Container(

width: MediaQuery.of(context).size.width,

height: 0.2,

color: Colors.grey,

),

);

}

return list;

}

\_getUsers() async {

\_rank = 0;

users.clear();

var conn = await MySqlConnection.connect(connect);

Results results = await (await conn

.execute('select userId,userName,num from user ORDER BY num DESC'))

.deStream();

results.forEach((element) {

setState(() {

\_flag = 1;

\_rank++;

int \_id = element.byName('userId');

String \_name = element.byName('userName');

int \_num = element.byName('num');

String \_medal;

if (\_num < 10) {

\_medal = '初进记账';

} else if (\_num < 100) {

\_medal = '停不下来';

} else if (\_num < 200) {

\_medal = '记账能手';

} else if (\_num < 300) {

\_medal = '记账小将';

} else if (\_num < 600) {

\_medal = '记账高手';

} else if (\_num < 1000) {

\_medal = '记账大师';

} else if (\_num < 2000) {

\_medal = '记账至尊';

} else {

\_medal = '高处不胜寒';

}

if (userId == \_id) {

\_use = User(\_id, \_name, \_num, \_medal);

\_use.rank = \_rank;

}

User \_user = User(\_id, \_name, \_num, \_medal);

users.add(\_user);

});

});

conn.close();

}

//初始化

@override

initState() {

super.initState();

\_getUsers();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

centerTitle: true,

backgroundColor: Color.fromARGB(255, 255, 200, 0),

iconTheme: IconThemeData(

color: Colors.black, //修改颜色

),

title: Text("排行榜", style: TextStyle(color: Colors.black)),

),

body: ListView(

children: \_body(),

),

);

}

}