**DOKUMEN DESAIN PERANGKAT LUNAK**

**L - PAY**

SOURCE CODE

*LSKK-SW-SC-01-L-PAY*



**PT. LANGGENG SEJAHTERA KREASI KOMPUTASI**

**©2019**

Dokumen Desain Perangkat Lunak, L-PAY

Source Code Sistem L-Pay

PT. Langgeng Sejahtera Kreasi Komputasi ©2019

# Deskripsi Aplikasi

Sistem L-Pay merupakan sistem pembayaran digital yang menggunakan QRCode. L-Pay dapat digunakan pada L-Commuter, L-Vending dan L-Parking. L-Commuter merupakan aplikasi yang berfungsi sebagai tiket digital yang digunakan pada saat masuk dan keluar stasiun kereta. L-Vending merupakan aplikasi yang berfungsi sebagai uang digital untuk melakukan transaksi pembelian pada mesin vending. L-Parking memiliki fungsi yang hampir mirip dengan L-Commuter dimana uang digital digunakan untuk masuk dan keluar parkiran dengan mengunakan QRCode. Untuk dapat mengisi saldo, L-Pay mempunyai sistem top-up. Sistem top-up memiliki 2(dua) metode pembayaran yaitu pembelian voucher dan transfer bank.

# *Source Code*

Berikut *source code system L-Pay*  :

Main.dart

|  |
| --- |
| import 'package:flutter/material.dart';  import 'package:lpay/RoutingConst.dart';  import 'package:lpay/ServiceLocator.dart';  import 'package:lpay/core/viewmodel/AppState.dart';  import 'package:provider/provider.dart';  import 'package:lpay/Router.dart' as router;  void main() {  setupLocator();  runApp(MyApp());  }  class MyApp extends StatelessWidget {  @override  Widget build(BuildContext context) {  return ChangeNotifierProvider(  builder: (context) => AppState(),  child: MainApplication(),  );  }  }  class MainApplication extends StatelessWidget {  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'L-Pay',  theme: ThemeData(fontFamily: 'Montserrat'),  onGenerateRoute: router.generateRoute,  initialRoute: OnBoardView,  );  }  } |

AppState.dart

|  |
| --- |
| import 'dart:convert';  import 'dart:io';  import 'dart:math';  import 'package:device\_info/device\_info.dart';  import 'package:flutter\_barcode\_scanner/flutter\_barcode\_scanner.dart';  import 'package:lpay/RoutingConst.dart';  import 'package:lpay/core/model/HistoryModel.dart';  import 'package:lpay/core/model/response/ResponseTicketModel.dart';  import 'package:mqtt\_client/mqtt\_client.dart' as mqtt;  import 'package:flutter/cupertino.dart';  import 'package:flutter/foundation.dart';  import 'package:flutter/material.dart';  import 'package:lpay/Const.dart';  import 'package:lpay/ServiceLocator.dart';  import 'package:lpay/core/service/ApiService.dart';  import 'package:lpay/core/service/MqttService.dart';  import 'package:lpay/core/service/StorageService.dart';  import 'package:progress\_dialog/progress\_dialog.dart';  import 'package:qr\_flutter/qr\_flutter.dart';  import 'package:rflutter\_alert/rflutter\_alert.dart';  import 'package:flutter\_money\_formatter/flutter\_money\_formatter.dart';  import 'package:fluttertoast/fluttertoast.dart';  class AppState with ChangeNotifier {  var apiService = locator<ApiService>();  var storageService = locator<StorageService>();  var mqttService = locator<MqttService>();  // Sign In  TextEditingController emailSignInController = TextEditingController();  // Sign Up  TextEditingController fullNameController = TextEditingController();  TextEditingController emailController = TextEditingController();  TextEditingController phoneNumberController = TextEditingController();  TextEditingController nominalTransferController = TextEditingController();  DeviceInfoPlugin deviceInfo = DeviceInfoPlugin();  ProgressDialog pr;  String deviceData;  String verificationCode;  String fullName;  String strBalance;  int balance;  QrImage qrImage;  QrImage qrImageTransfer;  HistoryModel dataHistory;  AppState() {  // connectMqtt();  \_getGuid();  }  // Connect to mqtt Server  Future connectMqtt() async {  await mqttService.connectAsync();  if (mqttService.client.connectionStatus.state ==  mqtt.MqttConnectionState.connected) {  var value = await storageService.loadStringValue(kGuidVal);  mqttService.deviceId = value;  mqttService.subscribeTopic();  } else {  /// Use status here rather than state if you also want the broker return code.  }  mqttService.client.updates.listen(  (List<mqtt.MqttReceivedMessage<mqtt.MqttMessage>> c) {  try {  final mqtt.MqttPublishMessage recMess = c[0].payload;  final String pt = mqtt.MqttPublishPayload.bytesToStringAsString(  recMess.payload.message);  print(  'MQTTCLASS::Change notification:: topic is <${c[0].topic}>, payload is <-- $pt -->');  var topic = c[0].topic;  if (topic.contains('user')) {  print('user mode');  var msg = json.decode(pt);  var lastBalance = msg['balance'];  var balanceInt = lastBalance.toInt();  var fullName = msg['fullname'];  storageService.saveStringValue(fullName, kFullNameVal);  storageService.saveIntValue(kBalanceVal, balanceInt);  updateProfile();  } else if (topic.contains('history')) {  var msg = pt.replaceAll("'", '"');  print(msg);  // var decode = json.decode(msg);  } else {  var msg = json.decode(pt);  switch (msg['tipe']) {  case 'request ticket':  final responseTicketModel = responseTicketModelFromJson(pt);  if (responseTicketModel.status == 'success') {  storageService.saveStringValue(  kTokenTicketVal, responseTicketModel.token);  generateTicketImage(responseTicketModel.token);  }  break;  case 'topup saldo':  if (msg['status'] == 'success') {  var lastBalance = msg['lastbalance'];  var balanceInt = lastBalance.toInt();  storageService.saveIntValue(kBalanceVal, balanceInt);  updateProfile();  Fluttertoast.showToast(  msg: "Topup Saldo Berhasil",  toastLength: Toast.LENGTH\_LONG,  gravity: ToastGravity.CENTER,  timeInSecForIos: 1,  backgroundColor: kColorBlue,  textColor: Colors.white,  fontSize: 16.0,  );  } else {  Fluttertoast.showToast(  msg: "Topup Saldo Gagal",  toastLength: Toast.LENGTH\_LONG,  gravity: ToastGravity.CENTER,  timeInSecForIos: 1,  backgroundColor: kColorBlue,  textColor: Colors.white,  fontSize: 16.0,  );  }  break;  case 'transfer with friends':  if (msg['status'] == 'success') {  var lastBalance = msg['lastbalance'];  var balanceInt = lastBalance.toInt();  storageService.saveIntValue(kBalanceVal, balanceInt);  updateProfile();  // Notification for successful balance update  Fluttertoast.showToast(  msg: "Transfer saldo berhasil",  toastLength: Toast.LENGTH\_LONG,  gravity: ToastGravity.CENTER,  timeInSecForIos: 1,  backgroundColor: kColorBlue,  textColor: Colors.white,  fontSize: 16.0,  );  }  break;  case 'receive with friends':  if (msg['status'] == 'success') {  var lastBalance = msg['lastbalance'];  var balanceInt = lastBalance.toInt();  storageService.saveIntValue(kBalanceVal, balanceInt);  updateProfile();  // Notification for successful balance update  Fluttertoast.showToast(  msg: "Saldo berhasil diterima",  toastLength: Toast.LENGTH\_LONG,  gravity: ToastGravity.CENTER,  timeInSecForIos: 1,  backgroundColor: kColorBlue,  textColor: Colors.white,  fontSize: 16.0,  );  }  break;  default:  print('nothin do here');  break;  }  }  notifyListeners();  } catch (e) {  print('[MQTT-MSGRCV] an error $e');  }  },  );  }  //  // Get GUID from device itself  //  Future \_getGuid() async {  try {  if (Platform.isAndroid) {  var androidInfo = await deviceInfo.androidInfo;  deviceData = androidInfo.androidId;  storageService.saveStringValue(kDeviceIdentifier, deviceData);  } else if (Platform.isIOS) {  var iosInfo = await deviceInfo.iosInfo;  deviceData = iosInfo.identifierForVendor;  storageService.saveStringValue(kDeviceIdentifier, deviceData);  }  } catch (x) {  print('[GetGuid] error with $x');  }  }  //  // Login method  //  Future doLogin(BuildContext context) async {  var email = emailSignInController.text;  var guid = deviceData;  if (email.isNotEmpty) {  pr.show();  var response = await apiService.loginService(email, guid);  if (response != null) {  pr.dismiss();  if (response.success) {  storageService.saveStringValue(kEmailVal, response.data.email);  storageService.saveStringValue(  kPhoneNumberVal, response.data.phonenumber);  storageService.saveIntValue(kBalanceVal, response.data.balance);  storageService.saveStringValue(kStatusVal, response.data.status);  storageService.saveStringValue(kTokenVal, response.data.token);  storageService.saveStringValue(kGuidVal, response.data.guid);  clearTextController();  return true;  } else {  showErrorAlert(context, 'Username atau Password salah');  return false;  }  } else {  pr.dismiss();  showErrorAlert(context, '${response.success}');  return false;  }  } else {  showErrorAlert(context, 'Harap isi field email');  return false;  }  }  //  // Verification Method  //  Future doVerification(BuildContext context) async {  var email = await storageService.loadStringValue(kEmailVal);  var token = verificationCode;  if (token.isNotEmpty) {  pr.show();  var response = await apiService.verificationService(email, token);  if (response != null) {  pr.dismiss();  if (response.success) {  storageService.saveStringValue(kIdVal, response.data[0].id);  storageService.saveStringValue(  kFullNameVal, response.data[0].fullname);  storageService.saveStringValue(kEmailVal, response.data[0].email);  storageService.saveStringValue(  kPhoneNumberVal, response.data[0].phonenumber);  storageService.saveIntValue(kBalanceVal, response.data[0].balance);  storageService.saveStringValue(kStatusVal, response.data[0].status);  storageService.saveStringValue(kTipeVal, response.data[0].tipe);  storageService.saveStringValue(kTokenVal, response.data[0].token);  storageService.saveStringValue(kGuidVal, response.data[0].guid);  connectMqtt();  clearTextController();  await updateProfile();  return true;  } else {  pr.dismiss();  showErrorAlert(context, 'Kode Verifikasi Salah');  return false;  }  } else {  pr.dismiss();  showErrorAlert(context, 'user tidak ditemukan');  return false;  }  } else {  pr.dismiss();  showErrorAlert(context, 'Harap isi field Verifikasi');  return false;  }  }  //  // Register Method  //  Future doRegister(BuildContext context) async {  var name = fullNameController.text;  var email = emailController.text;  var phone = phoneNumberController.text;  var guid = deviceData;  if (name.isNotEmpty && email.isNotEmpty && phone.isNotEmpty) {  pr.show();  var response = await apiService.registerService(name, email, phone, guid);  print('ini response $response');  if (response != null) {  pr.dismiss();  if (response.success) {  storageService.saveStringValue(kFullNameVal, name);  storageService.saveStringValue(kEmailVal, email);  storageService.saveStringValue(kPhoneNumberVal, phone);  storageService.saveStringValue(kGuidVal, guid);  clearTextController();  return true;  } else {  pr.dismiss();  showErrorAlert(context, 'Kode Verifikasi Salah');  return false;  }  } else {  pr.dismiss();  showErrorAlert(context, 'Email Sudah Terdaftar');  return false;  }  } else {  showErrorAlert(context, 'Harap isi field Verifikasi');  return false;  }  }  //  // Generate image for transfer balance  //  Future generateTransferImage(BuildContext context) async {  var id = await storageService.loadStringValue(kIdVal);  var guid = await storageService.loadStringValue(kGuidVal);  var currentBalance = await storageService.loadIntValue(kBalanceVal);  print('nominal ${nominalTransferController.text}');  if (nominalTransferController.text.isNotEmpty) {  var transferBalance = int.parse(nominalTransferController.text);  var currentIntBalance = currentBalance;  if (transferBalance > currentIntBalance) {  showErrorAlert(  context, 'Saldo anda tidak mencukupi untuk melakukan transaksi');  } else {  var jsonTransfer = {};  jsonTransfer['userTransferId'] = id;  jsonTransfer['balanceTransfer'] = nominalTransferController.text;  jsonTransfer['guid'] = guid;  jsonTransfer['msg\_type'] = 'transfer\_with\_friends';  var jsonString = json.encode(jsonTransfer);  print(jsonString);  qrImageTransfer = QrImage(  data: jsonString,  version: QrVersions.auto,  size: 320,  gapless: false,  errorStateBuilder: (cxt, err) {  return Container(  child: Center(  child: Text(  "Uh oh! Something went wrong...",  textAlign: TextAlign.center,  ),  ),  );  },  );  clearTextController();  notifyListeners();  }  } else {  showErrorAlert(context, 'Harap isi jumlah transfer');  }  }  //  // Generate image for ticketing  //  Future generateTicketImage(String token) async {  var id = await storageService.loadStringValue(kIdVal);  var guid = await storageService.loadStringValue(kGuidVal);  var jsonQr = {};  jsonQr['token'] = token;  jsonQr['myID'] = id;  jsonQr['guid'] = guid;  var jsonString = json.encode(jsonQr);  qrImage = QrImage(  data: jsonString,  version: QrVersions.auto,  size: 320,  gapless: false,  errorStateBuilder: (cxt, err) {  return Container(  child: Center(  child: Text(  "Uh oh! Something went wrong...",  textAlign: TextAlign.center,  ),  ),  );  },  );  return qrImage;  }  //  // Refresh Balance Method  //  Future refreshBalance() async {  try {  var id = await storageService.loadStringValue(kIdVal);  var guid = await storageService.loadStringValue(kGuidVal);  var msg = {};  msg['msg\_type'] = 'self\_qr';  msg['userReceiveId'] = id;  msg['guid'] = guid;  var jsonString = json.encode(msg);  mqttService.sendMessages(jsonString, 'request/user/$guid');  } catch (x) {  print('[refreshBalance] an error has ocurred with messages $e');  }  }  //  // Update UI  //  Future updateProfile() async {  try {  fullName = await storageService.loadStringValue(kFullNameVal);  balance = await storageService.loadIntValue(kBalanceVal);  strBalance = formatBalance(balance);  print(formatBalance(balance));  print(fullName);  notifyListeners();  } catch (x) {  print('[GetProfileUI] an error has ocurred with messages $e');  }  }  //  // Check if user has logged in  //  Future<bool> checkForLogin(BuildContext context) async {  initializeProgressHUD(context);  var value = await storageService.loadStringValue(kGuidVal);  print(value);  if (value != null) {  await updateProfile();  connectMqtt();  return true;  } else {  return false;  }  }  //  // Method for logout and clear all storage  //  Future logout(BuildContext context) async {  var value = await storageService.deleteAll();  if (value) {  Navigator.of(context).popUntil((route) => route.isFirst);  Navigator.pushReplacementNamed(context, OnBoardView);  mqttService.unsubscribeTopic();  mqttService.client.disconnect();  }  }  //  // Method for scan QrCode for topUp balance  //  Future<void> scanQR() async {  String barcodeScanRes;  // Platform messages may fail, so we use a try/catch PlatformException.  try {  barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(  "#ffff00", "Cancel", true, ScanMode.QR);  print(barcodeScanRes);  await topUpBalance(barcodeScanRes);  } catch (e) {  print('[scanQR] an error has ocurred with messages ${e.toString()}');  }  }  //  // Method for scanning other image  //  Future<void> transferQR() async {  String barcodeScanRes;  // Platform messages may fail, so we use a try/catch PlatformException.  try {  barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(  "#ffff00", "Cancel", true, ScanMode.QR);  print(barcodeScanRes);  var guid = await storageService.loadStringValue(kGuidVal);  var id = await storageService.loadStringValue(kIdVal);  var jsonDecoded = json.decode(barcodeScanRes);  var msg = {};  msg['msg\_type'] = 'transfer\_with\_friends';  msg['userTransferId'] = jsonDecoded['userTransferId'];  msg['userReceiveId'] = id;  msg['balanceTransfer'] = jsonDecoded['balanceTransfer'];  msg['guid'] = jsonDecoded['guid'];  msg['guidReceive'] = guid;  var msgStr = json.encode(msg);  mqttService.sendMessages(msgStr, 'request/transfer/$guid');  } catch (e) {  print('[scanQR] an error has ocurred with messages ${e.toString()}');  }  }  //  // TopUp balance from QrCode  //  Future topUpBalance(String strJson) async {  var id = await storageService.loadStringValue(kIdVal);  var guid = await storageService.loadStringValue(kGuidVal);  var decodedJson = json.decode(strJson);  try {  var serialNumber = decodedJson['serialNumber'];  var msgTopUp = {};  msgTopUp['msg\_type'] = 'topup';  msgTopUp['userReceiveId'] = id;  msgTopUp['serialNumber'] = serialNumber;  msgTopUp['guid'] = guid;  var jsonString = json.encode(msgTopUp);  print('jsonString $jsonString');  mqttService.sendMessages(jsonString, 'request/topup/$guid');  } catch (e) {  print('[TopUpBalance] an error has ocurred $e');  }  }  //  // Show success alert notification  //  void showSuccessAlert(BuildContext context, String msg) {  Alert(  context: context,  type: AlertType.success,  title: "SUCCESS",  desc: "$msg",  buttons: [  DialogButton(  child: Text(  "CLOSE",  ),  onPressed: () => Navigator.pop(context),  width: 120,  )  ],  closeFunction: () => Navigator.pop(context),  ).show();  }  //  // Show error notification  //  void showErrorAlert(BuildContext context, String msg) {  Alert(  context: context,  type: AlertType.error,  title: "WARNING",  desc: "$msg",  buttons: [  DialogButton(  child: Text(  "CLOSE",  ),  onPressed: () => Navigator.pop(context),  width: 120,  )  ],  closeFunction: () => Navigator.pop(context),  ).show();  }  //  // Show notification logout  //  void showNotificationLogout(BuildContext context) {  Alert(  context: context,  type: AlertType.warning,  title: "WARNING",  desc: "Do you really want to quit application ?",  buttons: [  DialogButton(  child: Text(  "YES",  style: TextStyle(color: Colors.white, fontSize: 20),  ),  onPressed: () async => await logout(context),  color: Color.fromRGBO(0, 179, 134, 1.0),  ),  DialogButton(  child: Text(  "NO",  style: TextStyle(color: Colors.white, fontSize: 20),  ),  onPressed: () => Navigator.pop(context),  color: Colors.red,  ),  ],  closeFunction: () {  Navigator.pop(context);  print('Close Function pressed');  },  ).show();  }  //  // Initialize progress HUD Method  //  void initializeProgressHUD(BuildContext context) {  print('Progress HUD being initialized');  pr = ProgressDialog(context);  }  //  // Requesting ticket from server  //  void requestTicket() async {  var guid = await storageService.loadStringValue(kGuidVal);  print(guid);  var msg = {};  msg['msg\_type'] = 'request\_ticket\_valid';  msg['guid'] = guid;  var msgEncoded = json.encode(msg);  print(msgEncoded);  print('request/ticket/$guid');  mqttService.sendMessages(msgEncoded, 'request/ticket/$guid');  }  //  // Clear all text controller (text field)  //  void clearTextController() {  try {  emailSignInController.clear();  fullNameController.clear();  emailController.clear();  phoneNumberController.clear();  nominalTransferController.clear();  } catch (e) {  print('[clearTextController] an error has ocurred [$e]');  }  }  //  // Format Balance  //  String formatBalance(int balance) {  var balanceFormatted = balance.toDouble();  FlutterMoneyFormatter fmf = FlutterMoneyFormatter(  amount: balanceFormatted,  settings: MoneyFormatterSettings(  symbol: 'Rp.',  thousandSeparator: '.',  decimalSeparator: ',',  symbolAndNumberSeparator: ' ',  fractionDigits: 2,  compactFormatType: CompactFormatType.short,  ),  );  return fmf.output.symbolOnLeft;  }  //  // Initialize progress HUD Method  //  Future getTransactionHistory() async {  try {  dataHistory = null;  var id = await storageService.loadStringValue(kIdVal);  var response = await apiService.historyService(id);  if (response != null) {  dataHistory = response;  notifyListeners();  }  } catch (e) {  print('[getTransactionHistory] an error has ocurred [$e]');  }  }  } |

# *Release Notes*

Berikut *release notes* aplikasi untuk sistem L-Pay.

### *L-PAY-SC, b1912, v1.0*

* Versi rilis pertama.

# *Approval*

Dibuat oleh Tim *S/W* *Engineer* pada 16 Desember 2019.

Dicek dan disetujui pada 16 Desember 2019, oleh:

Mochamad Vicky Ghani Aziz