

INGENIERÍA DE SOFTWARE

PROGRAMACIÓN EN DISPOSITIVOS MÓVILES

PROFESOR: JORGE ALBERTO HERNÁNDEZ TAPIA

ALUMNO: LUIS ARTURO ORTIZ SUAREZ

MATRICULA: 18100172

AI7.LOGIN PAGE

26/06/2022

Contenido

[INTRODUCCIÓN 2](#_Toc107180107)

[DESARROLLO 2](#_Toc107180108)

[VISTA 3](#_Toc107180109)

[CAMPOS VACÍOS 4](#_Toc107180110)

[LETRAS, NÚMEROS Y CORREO 4](#_Toc107180111)

[CÓDIGO MAIN 5](#_Toc107180112)

[CÓDIGO TEXTFIELD 9](#_Toc107180113)

[CÓDIGO TEXTFIELD CONTRASEÑA 10](#_Toc107180114)

[CONCLUSIÓN 11](#_Toc107180115)

[REPOSITORIO 11](#_Toc107180116)

# INTRODUCCIÓN

Para esta actividad, se realizará un login page utilizando Dart y flutter, dicha aplicación deberá realizar la validación para que todos los datos sean ingresados y que tengan un formato acorde.

# DESARROLLO

Para esta actividad lo primero que se realiza es hacer una personalización de los textfield donde se ingresaran los datos, en este sentido se crean 2 tipos:

* datos no sensibles: se pueden visualizar los caracteres que se están escribiendo
* contraseñas: no se visualizan los caracteres al momento de que se esta escribiendo sobre el campo, e utiliza la propiedad de ***obscureText: true.***

obscureText: true,

Otra de las características para textfield de datos no sensibles, es que en el constructor se le agregan 2 validadores, uno para que el campo no quede vacío y el otro para que el dato tenga cierto tipo de caracteres y así garantizar la integridad de la información.

const personalizarTextfield(  
 {Key? key,  
 required this.texto,  
 required this.miIcono,  
 required this.validator,  
 required this.validator2,  
 })  
 : super(key: key);

Una vez realizado esta personalización se procede a generar la pagina principal, donde agregamos nuestros campos y agregamos las funciones que validaran los campos, en este sentido se crean 4 funciones:

* no campos vacíos
* solo letras
* solo números
* validar que existe “@” en la cadena de caracteres

## VISTA

Las siguientes imágenes muestran el resultado final del desarrollo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

### CAMPOS VACÍOS

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

### LETRAS, NÚMEROS Y CORREO

Interfaz de usuario gráfica

Descripción generada automáticamente

## CÓDIGO MAIN

import 'package:flutter/material.dart';  
import 'package:flutter/widgets.dart';  
import 'estilostextfield.dart';  
import 'textfieldcontraseña.dart';  
import 'estilosboton.dart';  
  
void main() {  
 runApp(MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 debugShowCheckedModeBanner: false,  
 home: HomePage(),  
 theme: ThemeData.light(),  
 darkTheme: ThemeData.dark(),  
 );  
 }  
}  
  
class HomePage extends StatefulWidget {  
 const HomePage({Key? key}) : super(key: key);  
  
 @override  
 State<HomePage> createState() => \_HomePageState();  
}  
  
class \_HomePageState extends State<HomePage> {  
 final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();  
 final \_scaffoldKey = GlobalKey<ScaffoldState>();  
 TextEditingController contrasena1 = new TextEditingController();  
  
 //bool \_autovalidate = false;  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 key: \_scaffoldKey,  
 appBar: AppBar(  
 title: Text("AI7 Login page"),  
 centerTitle: true,  
 ),  
 body: ListView(  
 children: [  
 //mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 Column(  
 children: [  
 Center(  
 child: Container(  
 height: 120,  
 width: 120,  
 decoration: new BoxDecoration(  
 shape: BoxShape.circle,  
 image: new DecorationImage(  
 image: new NetworkImage(  
 "data:image/jpeg;base64,"))),  
 ),  
 ),  
 Form(  
 key: \_formKey,  
 autovalidateMode: AutovalidateMode.disabled,  
 child: Column(  
 //mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 personalizarTextfield(  
 texto: "Nombre",  
 miIcono: Icon(Icons.*person\_outline*),  
 validator: validacionNoVacio,  
 validator2: validarLetras),  
 personalizarTextfield(  
 texto: "Apellido paterno",  
 miIcono: Icon(Icons.*person\_pin\_outlined*),  
 validator: validacionNoVacio,  
 validator2: validarLetras),  
 personalizarTextfield(  
 texto: "Apellido materno",  
 miIcono: Icon(Icons.*person\_pin\_outlined*),  
 validator: validacionNoVacio,  
 validator2: validarLetras,  
 ),  
 personalizarTextfield(  
 texto: "Telefono",  
 miIcono: Icon(Icons.*phone\_outlined*),  
 validator: validacionNoVacio,  
 validator2: validarTelefono,  
 ),  
 personalizarTextfield(  
 texto: "Correo",  
 miIcono: Icon(Icons.*mail\_outline*),  
 validator: validacionNoVacio,  
 validator2: emailValidador,  
 ),  
 textfieldcontrasena(  
 texto: "Contraseña",  
 miIcono: Icon(Icons.*password*),  
 validator: validacionNoVacio),  
 textfieldcontrasena(  
 texto: "Confirmar contraseña",  
 miIcono: Icon(Icons.*password*),  
 validator: validacionNoVacio),  
 MaterialButton(  
 onPressed: () {  
 if (\_formKey.currentState!.validate()) {  
 //\_showSnackBar(context, "Informacion valida");  
 }  
 },  
 child: Text("enviar"),  
 color: Colors.*blue*,  
 )  
 ],  
 ),  
 ),  
 ],  
 ),  
 ],  
 ),  
 );  
 }  
  
 //motodos de validacion  
 String? validacionNoVacio(String? value) {  
 return value!.isEmpty ? "el campo no puede estar vacio" : null;  
 }  
  
 String? emailValidador(String? value) {  
 return value?.contains("@") == false ? "correo invalido" : null;  
 }  
  
 String? validarTelefono(String? value) {  
 final RegExp numberregexp = RegExp('[0-9]');  
 return numberregexp.hasMatch(value!) == false  
 ? "solo acepta numeros"  
 : null;  
 }  
  
 String? validarLetras(String? value) {  
 final RegExp numberregexp = RegExp('[a-z]');  
 return numberregexp.hasMatch(value!) == false ? "solo acepta letras" : null;  
 }  
}

## CÓDIGO TEXTFIELD

import 'package:flutter/material.dart';  
  
class personalizarTextfield extends StatelessWidget {  
 final String texto;  
 final Icon miIcono;  
 final FormFieldValidator<String> validator;  
 final FormFieldValidator<String> validator2;  
  
 const personalizarTextfield(  
 {Key? key,  
 required this.texto,  
 required this.miIcono,  
 required this.validator,  
 required this.validator2,  
 })  
 : super(key: key);  
 //personalizarTextfield(this.texto,this.miIcono, this.\_validator);  
 @override  
 Widget build(BuildContext context) {  
 return Container(  
 padding: EdgeInsets.only(left: 30.0, right: 30.0,top: 10,bottom: 10),  
 child: TextFormField(  
 validator: validator,  
 decoration: InputDecoration(  
 hintStyle: TextStyle(fontWeight: FontWeight.*bold*, fontSize: 20),  
 hintText: texto,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(30),  
 borderSide: BorderSide(  
 color: Colors.*blue*,  
 width: 2,  
 ),  
 ),  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(30),  
 borderSide: BorderSide(  
 color: Colors.*blue*,  
 width: 3,  
 ),  
 ),  
 prefixIcon: Padding(  
 padding: EdgeInsets.only(right: 10, left: 10),  
 child: miIcono,  
 )  
 ),  
 ),  
 );  
 }  
}

## CÓDIGO TEXTFIELD CONTRASEÑA

import 'package:flutter/material.dart';  
  
class textfieldcontrasena extends StatelessWidget {  
 final String texto;  
 final Icon miIcono;  
 final FormFieldValidator<String> validator;  
  
 const textfieldcontrasena(  
 {Key? key,  
 required this.texto,  
 required this.miIcono,  
 required this.validator})  
 : super(key: key);  
 //personalizarTextfield(this.texto,this.miIcono, this.\_validator);  
 @override  
 Widget build(BuildContext context) {  
 return Container(  
 padding: EdgeInsets.only(left: 30.0, right: 30.0,top: 10,bottom: 10),  
 child: TextFormField(  
 validator: validator,  
 decoration: InputDecoration(  
 hintStyle: TextStyle(fontWeight: FontWeight.*bold*, fontSize: 20),  
 hintText: texto,  
 enabledBorder: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(30),  
 borderSide: BorderSide(  
 color: Colors.*blue*,  
 width: 2,  
 ),  
 ),  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(30),  
 borderSide: BorderSide(  
 color: Colors.*blue*,  
 width: 3,  
 ),  
 ),  
 prefixIcon: Padding(  
 padding: EdgeInsets.only(right: 10, left: 10),  
 child: miIcono,  
 )  
 ),  
 obscureText: true,  
 ),  
 );  
 }  
}

# CONCLUSIÓN

Realizar validaciones dentro de una aplicación que recaba datos ayuda a garantizar la integridad de la información, sin embargo se deben conocer los diversos métodos que se emplean para lograr esto, desde validaciones simples hasta la utilización de expresiones regulares

# REPOSITORIO

https://github.com/laos89/pdm-ai7