Manual Técnico del Asilo La Esperanza en Santa Ana

Este manual técnico tiene como objetivo proporcionar una guía detallada de como se ha desarrollado la aplicación y así el personal encargado de administrar el sistema de citas médicas en el Asilo Esperanza de Santa Ana pueda tener conocimiento de la programación de la App. Asegurar que el sistema funcione de manera eficiente y que se brinde una atención médica oportuna y de calidad a los residentes del asilo.

Contenido

Inicio de Sesión

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(50, 24, 0, 0),
  child: InkWell(
    splashColor: Colors.transparent,
    focusColor: Colors.transparent,
    hoverColor: Colors.transparent,
    highlightColor: Colors.transparent,
    onTap: () async {
      setState(() {
        _model.create = false;
      });
    },
    child: Text(
      'Inicio de Sesión ',
      style: FlutterFlowTheme.of(context).displaySmall.override(
            fontFamily: 'Plus Jakarta Sans',
            color: _model.create == false
                ? Color(0xFF101213)
                : Color(0xFF57636C),
            fontSize: 36,
            fontWeight: FontWeight.w600,
          ),
    ),
  ),
)
```

se utiliza la clase Align para alinear un widget de texto en el centro del contenedor. La propiedad alignment toma un objeto de tipo AlignmentDirectional, que especifica la alineación del widget. AlignmentDirectional(0.5, -0.5) coloca el widget en el centro superior del contenedor.

```
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 16),
  child: Container(
    width: double.infinity,
    child: TextFormField(
      controller: _model.emailAddressController,
      focusNode: _model.emailAddressFocusNode,
      autofocus: true,
      autofillHints: [AutofillHints.email],
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Correo',
        labelStyle: FlutterFlowTheme.of(context).labelMedium.override(
              fontFamily: 'Plus Jakarta Sans',
              color: Color(0xFF57636C),
              fontSize: 14,
              fontWeight: FontWeight.w500,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFE0E3E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(40),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFF4B39EF),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(40),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
```

Registro de Usuario

La clase Expanded se utiliza generalmente para controlar cómo se distribuye el espacio dentro de un Row o Column. Por otro lado, EdgeInsetsDirectional se utiliza para gestionar el espacio en direcciones específicas, lo que es más útil para diseños que se ajustan a la dirección del texto, como en lenguajes escritos de derecha a izquierda.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.ingreseNombreController,
      focusNode: _model.ingreseNombreFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Nombre Completo',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintText: 'Ingrese el nombre completo',
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
```

Restablecer Contraseña

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

```
class CambioContraseaWidget extends StatefulWidget {
  const CambioContraseaWidget({Key? key}) : super(key: key);

  @override
  _CambioContraseaWidgetState createState() => _CambioContraseaWidgetState();
}

class _CambioContraseaWidgetState extends State<CambioContraseaWidget> {
  late CambioContraseaModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => CambioContraseaModel());

    _model.confirmarcontraseaController ??= TextEditingController();
    _model.confirmarcontraseaFocusNode ??= FocusNode();
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (isiOS) {
      SystemChrome.setSystemUIOverlayStyle(
```

Menú

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

```dart
class MenuWidget extends StatefulWidget {
  const MenuWidget({Key? key}) : super(key: key);

  @override
  _MenuWidgetState createState() => _MenuWidgetState();
}

class _MenuWidgetState extends State<MenuWidget> with TickerProviderStateMixin {
  late MenuModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => MenuModel());

    _model.searchBarController ??= TextEditingController();
    _model.searchBarFocusNode ??= FocusNode();

    _model.tabBarController = TabController(
      vsync: this,
      length: 3,
      initialIndex: 0,
    )..addListener(() => setState(() {}));
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
```

Opciones de Paciente

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

El método build toma un objeto BuildContext como argumento y devuelve la jerarquía de widgets que compondrán la interfaz de usuario de tu widget.

```dart
class OpcionesPacienteWidget extends StatefulWidget {
  const OpcionesPacienteWidget({Key? key}) : super(key: key);

  @override
  _OpcionesPacienteWidgetState createState() => _OpcionesPacienteWidgetState();
}

class _OpcionesPacienteWidgetState extends State<OpcionesPacienteWidget> {
  late OpcionesPacienteModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => OpcionesPacienteModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (isiOS) {
      SystemChrome.setSystemUIOverlayStyle(
        SystemUiOverlayStyle(
          statusBarBrightness: Theme.of(context).brightness,
          systemStatusBarContrastEnforced: true,
        ),
```

Registro de Paciente

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.ingreseNombreController,
      focusNode: _model.ingreseNombreFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Nombre Completo',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintText: 'Ingrese el nombre completo',
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
```

Editar o Eliminar Paciente

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.correoController,
      focusNode: _model.correoFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Nombre Completo',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
```

Opciones de Doctor

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

El método build toma un objeto BuildContext como argumento y devuelve la jerarquía de widgets que compondrán la interfaz de usuario de tu widget.

```dart
import 'opciones_doctor_model.dart';
export 'opciones_doctor_model.dart';

class OpcionesDoctorWidget extends StatefulWidget {
  const OpcionesDoctorWidget({Key? key}) : super(key: key);

  @override
  _OpcionesDoctorWidgetState createState() => _OpcionesDoctorWidgetState();
}

class _OpcionesDoctorWidgetState extends State<OpcionesDoctorWidget> {
  late OpcionesDoctorModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => OpcionesDoctorModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (isiOS) {
      SystemChrome.setSystemUIOverlayStyle(
        SystemUiOverlayStyle(
          statusBarBrightness: Theme.of(context).brightness,
          systemStatusBarContrastEnforced: true,
        ),
```

Registro de Doctor

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.ingreseNombreController,
      focusNode: _model.ingreseNombreFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Nombre Completo',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintText: 'Ingrese el nombre completo',
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
```

Editar o Eliminar Doctor

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.correoController,
      focusNode: _model.correoFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Nombre Completo',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
```

Opciones de Citas

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

El método build toma un objeto BuildContext como argumento y devuelve la jerarquía de widgets que compondrán la interfaz de usuario de tu widget.

```dart
class OpcionesCitasWidget extends StatefulWidget {
  const OpcionesCitasWidget({Key? key}) : super(key: key);

  @override
  _OpcionesCitasWidgetState createState() => _OpcionesCitasWidgetState();
}

class _OpcionesCitasWidgetState extends State<OpcionesCitasWidget> {
  late OpcionesCitasModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => OpcionesCitasModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (isiOS) {
      SystemChrome.setSystemUIOverlayStyle(
        SystemUiOverlayStyle(
          statusBarBrightness: Theme.of(context).brightness,
          systemStatusBarContrastEnforced: true,
        ),
```

Registro de Cita

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.ingreseNombreController,
      focusNode: _model.ingreseNombreFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'Nº Cita',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintText: 'Automatico',
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
```

Editar o Eliminar Cita

En esta clase EdgeInsets, se maneja los márgenes de manera estándar, start, top, end, y bottom son los valores de margen en las direcciones de inicio, arriba, final y abajo, respectivamente.

```
Expanded(
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(16, 8, 16, 8),
    child: TextFormField(
      controller: _model.correoController,
      focusNode: _model.correoFocusNode,
      obscureText: false,
      decoration: InputDecoration(
        labelText: 'N° Cita',
        labelStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        hintStyle: FlutterFlowTheme.of(context).bodyMedium.override(
              fontFamily: 'Overpass',
              color: FlutterFlowTheme.of(context).secondary,
            ),
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0xFFDBE2E7),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
            width: 2,
          ),
          borderRadius: BorderRadius.circular(8),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Color(0x00000000),
```

Historial

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

El método build toma un objeto BuildContext como argumento y devuelve la jerarquía de widgets que compondrán la interfaz de usuario de tu widget.

```dart
import 'historial_model.dart';
export 'historial_model.dart';

class HistorialWidget extends StatefulWidget {
  const HistorialWidget({Key? key}) : super(key: key);

  @override
  _HistorialWidgetState createState() => _HistorialWidgetState();
}

class _HistorialWidgetState extends State<HistorialWidget>
    with TickerProviderStateMixin {
  late HistorialModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  final animationsMap = {
    'textOnPageLoadAnimation': AnimationInfo(
      trigger: AnimationTrigger.onPageLoad,
      effects: [
        VisibilityEffect(duration: 100.ms),
        FadeEffect(
          curve: Curves.easeInOut,
          delay: 100.ms,
          duration: 600.ms,
          begin: 0,
          end: 1,
        ),
        MoveEffect(
          curve: Curves.easeInOut,
          delay: 100.ms,
          duration: 600.ms,
```

Configuraciones

La clase StatefulWidget se utiliza para gestionar el estado de un widget y permitir actualizaciones dinámicas en la interfaz de usuario. Son útiles cuando la parte de la interfaz de usuario en la que se encuentra describir puede cambiar dinámicamente.

El método build toma un objeto BuildContext como argumento y devuelve la jerarquía de widgets que compondrán la interfaz de usuario de tu widget.

```dart
class ConfiguracionesWidget extends StatefulWidget {
  const ConfiguracionesWidget({Key? key}) : super(key: key);

  @override
  _ConfiguracionesWidgetState createState() => _ConfiguracionesWidgetState();
}

class _ConfiguracionesWidgetState extends State<ConfiguracionesWidget> {
  late ConfiguracionesModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => ConfiguracionesModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (isiOS) {
      SystemChrome.setSystemUIOverlayStyle(
        SystemUiOverlayStyle(
          statusBarBrightness: Theme.of(context).brightness,
          systemStatusBarContrastEnforced: true,
        ),
```