TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ENSENADA

Programa de Ingeniería en Sistemas Computacionales

REPORTE

GI Vinification

Autores:
Itsai Zempoaltecatl Mejia
No Control: 16760300

Profesora:
ME Xenia Padilla Madrid

Programación de dispositivos moviles Android (enero-junio 2022)

Ensenada B.C. México
6 de junio de 2022

1. JUSTIFICACIÓN

API: **givinificationelretorno.appspot.com**

En el proceso de vinificación es necesario almacenar información de cada paso para la toma de desiciones o simplemente para llevar una bitacora. El proceso en la Vinicola El Retorno y en otras vinicolas este proceso es capturado en hojas de papel que a su vez son capturadas en un Excel.

Esto genera diferentes discrepancias en el seguimiento del proceso, es por esto que se pensó en un sistema en el que todos tenga acceso a la información, puedan agregar la misma y consultarla.

Se creó un API rest en el servicio de Firebase para el manejo de los datos a guardar y esta cuenta con diferentes tipos de colecciones.

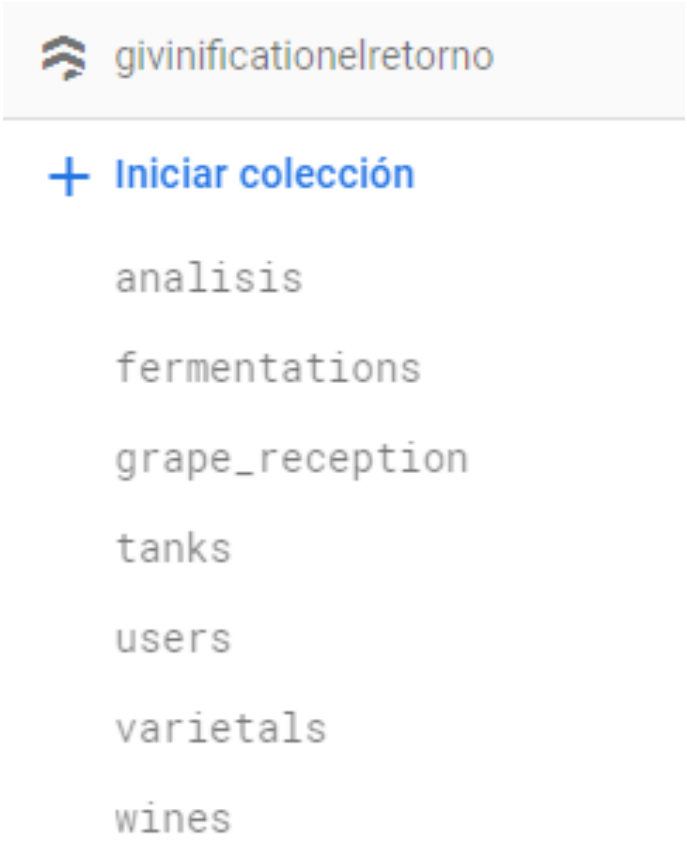

Figura 1: Colecciones de la API

2. PRUEBAS DE CONEXIÓN

givinificationelretorno.appspot.com /varietals

```dart
class VarietalServices extends ChangeNotifier {
  late CollectionReference collection;
  List<Varietal> listData = [];
  bool loadData = true;
  VarietalServices() {
    collection = FirebaseFirestore.instance.collection('varietals');
  }

  Future getList() async {
    try {
      QuerySnapshot snapshot = await collection.get();
      if (snapshot.size > 0) {
        listData = snapshot.docs.map((doc) {
          final result = Varietal.fromMap(doc.data() as Map<String, dynamic>);
          result.varietalId = doc.id.toDecodeId();
          return result;
        }).toList();
        print(listData);    Avoid `print` calls in production code.
      }
      loadData = false;
      notifyListeners();
    } catch (e) {
      loadData = false;
      notifyListeners();
    }
    loadData = false;
    notifyListeners();
  }
```

Figura 2: VarietalServices

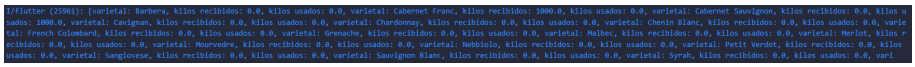

Figura 3: Data obtenida

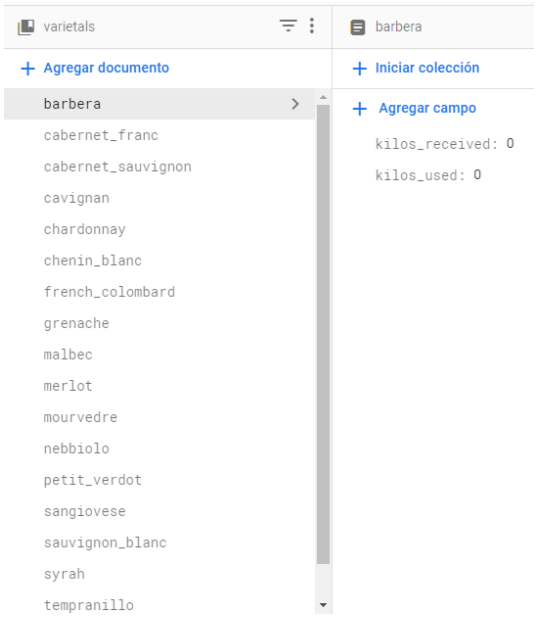

Figura 4: Colecciones de la base de datos

givinificationelretorno.appspot.com / tanks

```dart
class TankServices extends ChangeNotifier {
  late CollectionReference collection;
  List<Tank> listData = [];
  bool loadData = true;
  TankServices() {
    collection = FirebaseFirestore.instance.collection('tanks');
  }

  Future getList() async {
    try {
      QuerySnapshot snapshot = await collection.get();
      if (snapshot.size > 0) {
        listData = snapshot.docs.map((doc) {
          final result = Tank.fromMap(doc.data() as Map<String, dynamic>);
          result.tankId = doc.id.toDecodeId();
          return result;
        }).toList();
      }
      print(listData);    Avoid `print` calls in production code.
      loadData = false;
      notifyListeners();
    } catch (e) {
      loadData = false;
      notifyListeners();
    }
    loadData = false;
    notifyListeners();
  }
```

Figura 5: TankServices

```
I/flutter (25961): [Tanque 1, Cabernet Sauvignon-100, 500.0, Tinto, broth, Tanque 2, , 0.0, , , Tanque 3, , 0.0, , , Tanque 4, , 0.0, , ]
```

Figura 6: Data obtenida

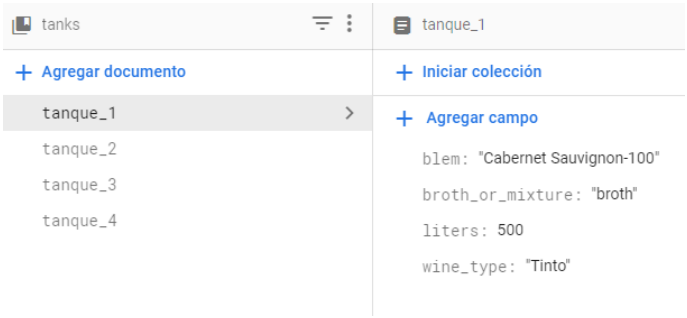

Figura 7: Colecciones de la base de datos

3. ADQUISICIÓN DE DATOS

MODELADO:

```dart
class Varietal {
  late String? varietalId;
  late double kilosReceived;
  late double kilosUsed;

  Varietal(
      {this.varietalId, required this.kilosReceived, required this.kilosUsed});

  Varietal.fromMap(Map<String, dynamic> map)
      : kilosReceived = double.parse(map['kilos_received'].toString()),
        kilosUsed = double.parse(map['kilos_used'].toString());

  Map<String, dynamic> toJson() =>
      {'kilos_received': kilosReceived, 'kilos_used': kilosUsed};
```

Figura 8: Clase Varietal

```dart
class Tank {
  Tank(
      {this.tankId,
      required this.blem,
      required this.liters,
      required this.wineType,
      required this.brothOrMixture});

  final String blem;
  final double liters;
  final String wineType;
  String? tankId;
  final String brothOrMixture;

  factory Tank.fromMap(Map<String, dynamic> map) => Tank(
      blem: map['blem'],
      liters: double.parse(map["liters"].toString()),
      wineType: map['wine_type'],
      brothOrMixture: map['broth_or_mixture']);

  Map<String, dynamic> toJson() => {
        "blem": blem,
        "liters": liters,
        'wine_type': wineType,
        'broth_or_mixture': brothOrMixture
      };
```

Figura 9: Clase Tank

```dart
class DataUser {
  late String? uid;
  final String name;
  final String surnames;
  final String role;
  final String email;

  DataUser(
      {this.uid,
      required this.name,
      required this.surnames,
      required this.role,
      required this.email});

  DataUser.fromMap(Map<String, dynamic> map)
      : name = map['name'],
        surnames = map['surnames'],
        role = map['role'],
        email = map['email'];

  Map<String, dynamic> toJson() =>
      {'name': name, 'surnames': surnames, 'email': email, 'role': role};
```

Figura 10: Clase DataUser

```dart
class GrapeReception {
  final String date;
  late String? id;
  final String varietal;
  final String ranch;
  final double brix;
  final double ph;
  final double kilos;
  final String responsible;

  GrapeReception(
      {required this.date,
      this.id,
      required this.varietal,
      required this.ranch,
      required this.brix,
      required this.ph,
      required this.kilos,
      required this.responsible});

  GrapeReception.fromMap(Map<String, dynamic> map)
      : date = map['date'],
        varietal = map['varietal'],
        ranch = map['ranch'],
        brix = double.parse(map['brix'].toString()),
        ph = double.parse(map['ph'].toString()),
        kilos = double.parse(map['kilos'].toString()),
        responsible = map['responsible'];

  Map<String, dynamic> toJson() => {
        'date': date,
        'varietal': varietal,
        'ranch': ranch,
        'brix': brix,
        'ph': ph,
        'kilos': kilos,
        'responsible': responsible
      };
```

Figura 11: Clase Grape Reception

```dart
class Analisis {
  final String date;
  late String? id;
  final String type;
  final double brix;
  final double ph;
  final String analysisPerformed;
  final String observations;
  final String itemAnalyzed;
  final String receptionOrTank;
  final String responsible;

  Analisis(
      {required this.date,
      this.id,
      required this.type,
      required this.brix,
      required this.ph,
      required this.analysisPerformed,
      required this.observations,
      required this.itemAnalyzed,
      required this.receptionOrTank,
      required this.responsible});

  Analisis.fromMap(Map<String, dynamic> map)
      : date = map['date'],
        type = map['type'],
        brix = double.parse(map['brix'].toString()),
        ph = double.parse(map['ph'].toString()),
        analysisPerformed = map['analysis_performed'],
        observations = map['observations'],
        itemAnalyzed = map['item_analyzed'],
        receptionOrTank = map['reception_or_tank'],
        responsible = map['responsible'];

  Map<String, dynamic> toJson() => {
        'date': date,
        'type': type,
        'brix': brix,
        'ph': ph,
        'analysis_performed': analysisPerformed,
        'observations': observations,
        'item_analyzed': itemAnalyzed,
        'reception_or_tank': receptionOrTank,
        'responsible': responsible
      };
```

Figura 12: Clase Analisis

```dart
class Fermentation {
  final String date;
  late String? id;
  final String activity;
  final int time;
  final String responsible;
  final String observations;
  final String whoMade;

  Fermentation(
      {required this.date,
      this.id,
      required this.activity,
      required this.time,
      required this.responsible,
      required this.observations,
      required this.whoMade});

  Fermentation.fromMap(Map<String, dynamic> map)
      : date = map['date'],
        activity = map['activity'],
        time = map['time'],
        responsible = map['responsible'],
        observations = map['observations'],
        whoMade = map['who_made'];

  Map<String, dynamic> toJson() => {
        'date': date,
        'activity': activity,
        'time': time,
        'responsible': responsible,
        'observations': observations,
        'who_made': whoMade
      };
```

Figura 13: Clase Fermentation

```dart
class Wine {
  late String? id;
  final String date;
  final String type;
  final double liters;
  final String anada;
  final String tankName;
  final String ranch;
  final String observations;
  late String blem;
  final String responsible;

  Wine(
      {this.id,
      required this.blem,
      required this.date,
      required this.type,
      required this.anada,
      required this.tankName,
      required this.liters,
      required this.ranch,
      required this.observations,
      required this.responsible});

  Wine.fromMap(Map<String, dynamic> map)
      : date = map['date'],
        type = map['type'],
        blem = map['blem'],
        liters = double.parse(map["liters"].toString()),
        anada = map['anada'],
        tankName = map['tank_name'],
        ranch = map['ranch'],
        observations = map['observations'],
        responsible = map['responsible'];

  Map<String, dynamic> toJson() => {
        'date': date,
        'type': type,
        'blem': blem,
        'liters': liters,
        'tank_name': tankName,
        'ranch': ranch,
        'anada': anada,
        'observations': observations,
        'responsible': responsible
      };
```

Figura 14: Clase Wine

SERVICES:
En los Services la lógica aplicada al inicio es la misma para casi todos y lo único que varia es el parametro de **FirebaseFirestore.instance.collection('<colección>');**
Ejemplos:

```
late CollectionReference collection;
List<Analisis> listData = [];
bool loadData = true;
AnalisisServices() {
  collection = FirebaseFirestore.instance.collection('analisis');
}
```

Figura 15: Variables públicas de AnalisisServices y su constructor

```
late CollectionReference collection;
List<GrapeReception> listData = [];
bool loadData = true;
GrapeReceptionServices() {
  collection = FirebaseFirestore.instance.collection('grape_reception');
}
```

Figura 16: Variables públicas de GrapeReceptionServices y su constructor

Métodos para obtener los datos
En cuanto a los métodos pasa algo similar, la manera de obtener y enviar los datos solo va cambiando el tipo de dato obtenido o enviado (Analisis, GrapeReception, Fermentation, etc)

```
Future getList() async {
  try {
    QuerySnapshot snapshot = await collection.get();

    if (snapshot.size > 0) {
      listData = snapshot.docs.map((doc) {
        final result =
            GrapeReception.fromMap(doc.data() as Map<String, dynamic>);
        result.id = doc.id;
        return result;
      }).toList();
    }
    loadData = false;
    notifyListeners();
  } catch (e) {
    loadData = false;
    notifyListeners();
  }
  loadData = false;
  notifyListeners();
}
```

Figura 17: getList() Clase GrapeReception

QuertSnapshot almacena los datos obtenidos del metodo get de la colección seleccionada, este contiene diferentes propiedades las cuales son utiles para verificar y asignar los datos. Se verifica que la consulta no este vacía para después asignar a listData la lista de los datos obtenidos después de modelar el resultado con el método fromMap de nuestra clase. Una vez asignada la información a listData se notifica a los listeners que la consulta concluyo con loadData = false.

```dart
Future get(String id) async {
  try {
    final snapshot = await collection.doc(id.toEncodeId()).get();
    if (snapshot.exists) {
      final response = Tank.fromMap(snapshot.data() as Map<String, dynamic>);
      response.tankId = snapshot.id.toDecodeId();
      return response;
    } else {
      return null;
    }
  } catch (e) {
    return null;
  }
}
Future<bool> add(Tank data) async {
  try {
    await collection.doc(data.tankId?.toEncodeId()).set(data.toJson());
    return true;
  } catch (e) {
    return false;
  }
}
Future<bool> update(Tank data) async {
  try {
    await collection.doc(data.tankId?.toEncodeId()).update(data.toJson());
    return true;
  } catch (e) {
    return false;
  }
}
Future<bool> delete(String id) async {
  try {
    await collection.doc(id.toEncodeId()).delete();
    return true;
  } catch (e) {
    return false;
  }
}
```

Figura 18: CRUD Tanks

```dart
Future get(String id) async {
  try {
    final snapshot = await collection.doc(id).get();
    return snapshot.exists
        ? GrapeReception.fromMap(snapshot.data() as Map<String, dynamic>)
        : null;
  } catch (e) {
    return null;
  }
}
Future<bool> add(GrapeReception data) async {
  try {
    await collection.doc(data.id).set(data.toJson());
    return true;
  } catch (e) {
    return false;
  }
}
Future<bool> update(GrapeReception data) async {
  try {
    await collection.doc(data.id).update(data.toJson());
    return true;
  } catch (e) {
    return false;
  }
}
Future<bool> delete(String id) async {
  try {
    await collection.doc(id).delete();
    return true;
  } catch (e) {
    return false;
  }
}
```

Figura 19: CRUD Grape Reception

get(id) obtiene una colección especifíca filtrada por ID del registro.
add(data) agrega el modelo obtenido por parametro en la colección.
update(data) actualiza los datos del documento que concida con el ID del modelo obtenido por parametro.
delete(id) elimina el documento en base al ID obtenido por parametro.

4. MOSTRAR DATOS AL USUARIO

La aplicación se divide en secciones, para acceder a cada una de ellas tenemos un menu inferior que contiene 4 opciones para acceder a cada pantalla.

```
return Scaffold(
  appBar: AppBar(
      automaticallyImplyLeading: false,
      elevation: 1,
      backgroundColor: primaryColor,
      title: Text(pages[indexPage]['title']),
      actions: [
        ActionsUserButton(name: dataUser.name, popupsMenuItems: [
          if (dataUser.role == 'Admin')
            PopupMenuItem(
                height: 40,
                onTap: () => setState(() => indexPage = 5),
                child: const Text(
                  "Administrar",
                  style: TextStyle(color: Colors.white, fontSize: 15),
                )), // Text // PopupMenuItem
          PopupMenuItem(
              height: 40,
              onTap: onSingOut,
              child: const Text(
                "Salir",
                style: TextStyle(color: Colors.white, fontSize: 15),
              )), // Text // PopupMenuItem
        ]) // ActionsUserButton
      ]), // AppBar
  body: Container(color: backgroundColor, child: pages[indexPage]['page']),
  bottomNavigationBar: Container(
    height: 55,
    padding: const EdgeInsets.only(bottom: 4),
    color: primaryColor,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        BottomNavigationBarItemCustom(
          permission: true,
          colorItem: indexPage == 0 ? Colors.white : secundaryColor,
          onTap: () => setState(() => indexPage = 0),
          icon: Icons.dashboard,
        ), // BottomNavigationBarItemCustom
        BottomNavigationBarItemCustom(
          permission: dataUser.role != 'Sin rol',
          colorItem: indexPage == 1 ? Colors.white : secundaryColor,
          onTap: () => setState(() => indexPage = 1),
          icon: Icons.receipt,
        ), // BottomNavigationBarItemCustom
        BottomNavigationBarItemCustom(
          permission: dataUser.role == 'Admin' ||
              dataUser.role == 'Enólogo' ||
              dataUser.role == 'Analista',
          colorItem: indexPage == 2 ? Colors.white : secundaryColor,
          icon: Icons.assignment,
          onTap: () => setState(() => indexPage = 2),
        ), // BottomNavigationBarItemCustom
        BottomNavigationBarMultItems(
          permission: dataUser.role == 'Admin' ||
              dataUser.role == 'Enólogo' ||
              dataUser.role == 'Operador',
          popupsMenuItems: [
            PopupMenuItem(
                height: 20,
                onTap: () => setState(() => indexPage = 3),
                child: const Text(
                  "Fermentaciones",
                  style: TextStyle(color: Colors.white, fontSize: 15),
                )), // Text // PopupMenuItem
            PopupMenuItem(
                height: 20,
                onTap: () => setState(() => indexPage = 4),
                child: const Text(
                  "Vino",
                  style: TextStyle(color: Colors.white, fontSize: 15),
                )), // Text // PopupMenuItem
          ],
          icon: Icons.wine_bar,
          colorItem: indexPage == 3 || indexPage == 4
              ? Colors.white
              : secundaryColor,
        ) // BottomNavigationBarMultItems
      ],
    ), // Row
  ), // Container
); // Scaffold
```

Figura 20: HomePage

Cada pantalla cuenta con widgets que comparten, los cuales son el botón de agregar, cuadro de texto para buscar por id y un seleccionador para filtrar por fecha. Con estos widgets el usuario puede interactuar con todos los datos.

```
body: Container(
    padding: EdgeInsets.only(
        left: mediaWidth > 820 ? 10 : 0,
        right: mediaWidth > 820 ? 10 : 0), // EdgeInsets.only
    height: double.infinity,
    child: Table(
      defaultVerticalAlignment: TableCellVerticalAlignment.middle,
      children: [
        TableRow(children: [
          Container(
              height: mediaWidth > 820 ? 99 : 60,
              padding:
                  const EdgeInsets.only(top: 10, left: 10, right: 10),
              child: Row(
                children: [
                  if (mediaWidth > 820)
                    Row(
                      children: [
                        AddButton(
                          onTap: onAdd,
                          onEnter: (event) => setState(
                              () => isMouseEnterButtonAdd = true),
                          onExit: (event) => setState(
                              () => isMouseEnterButtonAdd = false),
                          isMouseEnter: isMouseEnterButtonAdd,
                        ), // AddButton
                        const SizedBox(
                          width: 15,
                        ), // SizedBox
                      ],
                    ), // Row
                  InputSearch(
                    onTap: stateInput,
                    fontSize: mediaWidth > 820 ? 15 : 12,
                    iconSize: mediaWidth > 820 ? 23 : 18,
                    sizableState: sizableStateInput,
                    maxWidth: 200,
                    tooltipMessage: 'Buscar por o varietal',
                    labelText: "ID/Varietal",
                    controller: inputController,
                    onSubmitted: (value) {
                      if (value.toUpperCase().isAnID) {
                        filterByID(value);
                      } else if (value.isAName) {
                        filterByVarietal(value);
                      } else {
                        filterNoData(value);
                      }
                    },
                  ), // InputSearch
                  const SizedBox(
                    width: 15,
                  ), // SizedBox
                  InputDate(
                      iconSize: mediaWidth > 820 ? 23 : 18,
                      fontSize: mediaWidth > 820 ? 15 : 12,
                      sizableState: sizableStateDatePicker,
                      labelText: dateController.text,
                      onTap: filterByDate), // InputDate
                  if (mediaWidth < 820) const Spacer(),
                  ClearFiltersButton(
                      size: mediaWidth > 820 ? 20 : 15,
                      onPressed: clearFilters), // ClearFiltersButton
                  if (mediaWidth > 820) const Spacer(),
                  if (mediaWidth > 820)
                    const SizedBox(
                      width: 15,
                    ), // SizedBox
                ],
              )), // Row // Container
        ]), // TableRow
        const TableRow(children: [
          Divider(
            color: secundaryColor,
          ) // Divider
        ]), // TableRow
        TableRow(children: [
          SizedBox(
            height: mediaHeight - 170,
            child: SingleChildScrollView(
                primary: false,
                scrollDirection: Axis.vertical,
                child: buildTable()), // SingleChildScrollView
          ) // SizedBox
        ]) // TableRow
      ],
    )), // Table // Container
```

Figura 21: GrapeReceptionPage

A su vez cada pantalla cuenta con una tabla donde se muestran todos los datos obtenidos de las consultas y filtraciones.

```
buildTable() {
  List<DataCell> getCells(List<dynamic> cells) => cells
      .map((data) => DataCell(
            Material(
                type: MaterialType.transparency,
                child: Text('$data',
                    style: TextStyle(fontSize: mediaWidth > 820 ? 16 : 11))),
          )) // DataCell
      .toList();

  List<DataRow> getRows(List<GrapeReception> dataList) =>
      dataList.map((GrapeReception data) {
        final cells = [
          data.date,
          data.id,
          data.varietal,
          data.ranch,
          data.brix,
          data.ph,
          data.kilos,
        ];

        return DataRow(
            color: MaterialStateColor.resolveWith((states) {
              return contentTableColor;
            }),
            cells: getCells(cells) +
                [
                  DataCell(
                    ActionsIndexTable(
                      enabled: !(data.id == 'no data'),
                      iconSize: mediaWidth < 820 ? 16 : 25,
                      userType: Globals.getUserRole(),
                      onEdited: () async {
                        try {
                          final response = await showEditForm(
                              context: context,
                              content: FormEditGrapeReception(
                                grapeReception: data,
                                grapeReceptionServices:
                                    grapeReceptionServices,
                                varietalServices: varietalServices,
                                varietalsName: listVarietalsName,
                              )); // FormEditGrapeReception
                          if (response) {
                            await reloadData();
                          }
                        } catch (e) {}
                      },
                      onDeleted: () async {
                        try {
                          final response = await showDeleteForm(
                              context: context,
                              content: FormDeleteGrapeReception(
                                grapeReception: data,
                                grapeReceptionServices:
                                    grapeReceptionServices,
                                varietalServices: varietalServices,
                              )); // FormDeleteGrapeReception
                          if (response) {
                            await reloadData();
                          }
                        } catch (e) {}
                        await reloadData();
                      },
                      onViewResponsible: () {
                        showDialog(
                            context: context,
                            builder: (__) => AlertDialog(
                                  title: const Text('Responsable:'),
                                  content: Text(data.responsible),
                                )); // AlertDialog
                      },
                    ), // ActionsIndexTable
                  ) // DataCell
                ]); // DataRow
      }).toList();
  return DataTable(
      horizontalMargin: 10,
      headingRowColor:
          MaterialStateColor.resolveWith((states) => headerTableColor),
      columnSpacing: mediaWidth > 820 ? 10 : 4,
      headingRowHeight: mediaWidth > 820 ? 45 : 25,
      showCheckboxColumn: false,
      columns: List<DataColumn>.generate(
          GrapeReception.titles.length,
          (index) => DataColumn(
              label: Material(
                  type: MaterialType.transparency,
                  child: Text(GrapeReception.titles[index],
                      style: TextStyle(
                        color: backgroundColor,
                        fontSize: mediaWidth > 820 ? 20 : 11,
                        fontWeight: FontWeight.bold,
                      ))))), // TextStyle // Text // Material // DataColumn /
      rows: getRows(listGrapeReception)); // DataTable
```

Figura 22: GrapeReceptionPage Table

Visualización

Figura 23: GrapeReceptionPage resultado (información obtenida)

Nota: La construcción de cada pantalla es igual una a otra, lo único que va cambiando son los titulos de las tablas y la información que se muestra como anteriormente ya se explico cómo se obtiene en base al Modelo

5. REFERENCIAS

Firebase. (2022). Primeros pasos con Cloud Firestore. Firebase.
https://firebase.google.com/docs/firestore/quickstart
Firebase. (2022). Comience con la autenticación de Firebase en Flutter. Firebase.
https://firebase.google.com/docs/auth/flutter/start
Xenia Padilla.(2022). 53 FLUTTER REST API - FIREBASE. Youtube. [Video]
https://www.youtube.com/watch?v=AfzgqGjQzGI
Flutter. (2022). DataTable class. https://api.flutter.dev/flutter/material/DataTable-class.html
NARESH PRADEEP. (2021). Flutter Datepicker Widget Example Tutorial. CODES INSIDER.
https://codesinsider.com/flutter-datepicker-widget-example-tutorial/