Desis	**REQUERIMIENTO TÉCNICO**

1.- IDENTIFICACIÓN DEL REQUERIMIENTO			
Proyecto	Facturacion.cl		
Folio	Actividad 2	**Versión**	1.0
Nombre	Crear interfaz que permita registrar usuarios a través de un formulario. Debe haber un botón que obtenga los datos de una API y modifique los campos del formulario.		
Nº de entrega	1		

2.- DATOS DE ENTREGA	
Fecha Entrega	23-10-2024
Nombre Analista	Nicolás Hidalgo
Observación	

Crear Archivo en Flutter			
Acción	- Crear Archivo - Crear Función	***Nombre***	register_screen

Como se indica en la imagen 1:

- Se debe crear una vista que permita registrar una cuenta según los parámetros solicitados en un formulario.

Definición de parámetros a recibir o utilizar:

- BuildContext context
- TextEditingController _nameController
- TextEditingController _emailController
- TextEditingController _passwordController
- TextEditingController _addressController
- TextEditingController _dateController
- TextEditingController _phoneNumber

Definición de lógica a realizar

- **Validar los campos de texto:** Los campos del formulario deben estar validados según las reglas establecidas.
- **Registrar datos en una base de datos:** El usuario creado debe guardarse en una base de datos local para su posterior visualización.
- **Realizar consultas hacia una API externa:** Al presionar el botón de obtener usuario se deben obtener los datos aleatorios de un usuario desde una API para luego mostrar sus datos en los campos de texto.

IMAGEN 1

```
class RegisterScreenState extends State<RegisterScreen> {
  Widget build(BuildContext context) {
                label: "Nombre Completo",
                validator: (value) =>
                    validators.nameValidator(context, value)), // CustomTextFormField
            CustomTextFormField(
                controller: _emailController,
                label: "Correo Electrónico",
                validator: (value) =>
                    validators.emailValidator(value, context)), // CustomTextFormField
            CustomTextFormField(
                controller: _addressController,
                label: "Dirección",
                validator: validators.addressValidator), // CustomTextFormField
            CustomTextFormField(
                controller: _phoneNumber,
                label: "Número de Teléfono",
                validator: validators.phoneNumberValidator), // CustomTextFormField
            CustomTextFormField(
                controller: _dateController,
                label: "Fecha de Nacimiento",
                onTap: _onSelectedDate,
                readOnly: true,
                validator: validators.birthdayValidator), // CustomTextFormField
            CustomTextFormField(
                controller: _passwordController,
                label: "Contraseña",
                validator: validators.passwordValidator), // CustomTextFormField
            const SizedBox(height: 30),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 10.0),
              child: CustomButton(
                  onPress: () {
                    _formKey.currentState!.save();
                    if (_formKey.currentState!.validate()) {
                      handleRegister();
                    }
                  },
                  isDisabled: !isValid,
                  label: "Registrar Usuario",
                  width: MediaQuery.of(context).size.width), // CustomButton
            ), // Padding
            const SizedBox(
              height: 5,
            ), // SizedBox
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 10.0),
              child: CustomButton(
                onPress: () {
                  handleGetUser();
                },
                label: "Obtener desde API",
                width: MediaQuery.of(context).size.width,
                isLoading: isLoading,
                isDisabled: isLoading,
```

Requerimiento ####

IMAGEN 2

```dart
String? emailValidator(String? value) {
  if (!value!.contains('@') || !value.contains('.')) {
    return 'Por favor ingrese un correo válido';
  }
  if (value.contains(' ')) {
    return 'El correo no puede contener espacios en blanco';
  }
  return null;
}

String? nameValidator(String? value) {
  if (value == null || value.isEmpty) {
    return 'El campo nombre es obligatorio';
  }
  if (value.length < 2 || value.length > 50) {
    return 'El nombre debe rondar entre 2 y 50 caracteres';
  }
  if (!value.contains(RegExp(r"^[a-zA-ZáéíóúÁÉÍÓÚñÑ\s]+$"))) {
    return 'El nombre solo debe contener letras';
  }
  return null;
}

String? addressValidator(String? value) {
  if (value!.length < 5 || value.length > 100) {
    return 'La dirección debe rondar entre 5 y 100 caracteres';
  }
  return null;
}

String? birthdayValidator(String? value) {
  final now = DateTime.now();

  if (!value!.contains(RegExp(r'^\d{4}-\d{2}-\d{2}$'))) {
    return 'Ingrese una fecha válida en el formato YYYY-MM-DD';
  }
  if (DateTime.parse(value).isAfter(DateTime.now()) ||
      DateTime.parse(value)
          .isAtSameMomentAs(DateTime(now.year, now.month, now.day))) {
    return 'La fecha debe ser anterior';
  }

  return null;
}

String? passwordValidator(String? value) {
  if (value!.length < 8) {
    return 'La contraseña debe tener al menos 8 caracteres';
  }
  if (!RegExp(r'(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$ %^&*-]).{8,}')
      .hasMatch(value)) {
    return 'debe tener al menos una mayúscula, numero y caracter especial';
  }
  return null;
}
```

Crear Función en Flutter			
Acción	- Crear función	***Nombre***	fetchUser() || handleGetUser()

Requerimiento ####

Definición de parámetros a recibir o utilizar:

- String apiUrl

Definición de lógica a realizar

- **Consulta API:** Se debe establecer conexión con una API que obtiene datos de un usuario aleatorio
- **Agregar datos al formulario:** Al obtener los datos se deben modificar los campos de texto según los datos obtenidos.

IMAGEN 1

```dart
Future<User?> fetchUser() async {
  final response = await http.get(Uri.parse(apiUrl));

  if (response.statusCode == 200) {
    final json = jsonDecode(response.body);
    final userJson = json['results'][0];
    return User.fromJson(userJson);
  }
  return null;
}
```

IMAGEN 2

```dart
void handleGetUser() async {
  setIsLoading(true);
  final user = await userService.fetchUser();
  _nameController.text = user!.name;
  _emailController.text = user.email;
  _addressController.text = user.address;
  _dateController.text = user.birthDate.split('T')[0];
  _phoneNumber.text = user.phoneNumber.replaceAll(RegExp(r'[^0-9]'), '');
  _passwordController.text = user.password;
  setIsLoading(false);
}
```

Crear Archivo en Flutter			
Acción	- Crear archivo	***Nombre***	user

Definición de parámetros a recibir o utilizar:

- int id
- String name
- String email
- String birthDate
- String address
- String phoneNumber
- String password

Requerimiento ####

Definición de lógica a realizar

- **Modelo de usuario:** Se debe crear una Entidad de un usuario para poder gestionar sus datos.

IMAGEN 1

```dart
class User {
  final int? id;
  final String name;
  final String email;
  final String birthDate;
  final String address;
  final String phoneNumber;
  final String password;

  User({
    this.id,
    required this.name,
    required this.email,
    required this.birthDate,
    required this.phoneNumber,
    required this.address,
    required this.password,
  });

  factory User.fromJson(Map<String, dynamic> json) {
    String fullName = '${json['name']['first']} ${json['name']['last']}';
    String jsonAddress = '${json['location']['country']}';
    String birthDate = json['dob']['date'];

    return User(
      id: null,
      name: fullName,
      email: json['email'],
      birthDate: birthDate,
      phoneNumber: json['phone'],
      address: jsonAddress,
      password: json['login']['password'],
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'name': name,
      'email': email,
      'phoneNumber': phoneNumber,
      'birthDate': birthDate,
      'address': address,
      'password': password,
    };
  }
}
```

Crear Archivo en Flutter

Acción	- Crear archivo	***Nombre***	user

Definición de parámetros a recibir o utilizar:

- int id
- String name
- String email
- String birthDate
- String address
- String phoneNumber
- String password

Requerimiento ####

Definición de lógica a realizar

- **Modelo de usuario:** Se debe crear una Entidad de un usuario para poder gestionar sus datos.

IMAGEN 1

```dart
class User {
  final int? id;
  final String name;
  final String email;
  final String birthDate;
  final String address;
  final String phoneNumber;
  final String password;

  User({
    this.id,
    required this.name,
    required this.email,
    required this.birthDate,
    required this.phoneNumber,
    required this.address,
    required this.password,
  });

  factory User.fromJson(Map<String, dynamic> json) {
    String fullName = '${json['name']['first']} ${json['name']['last']}';
    String jsonAddress = '${json['location']['country']}';
    String birthDate = json['dob']['date'];

    return User(
      id: null,
      name: fullName,
      email: json['email'],
      birthDate: birthDate,
      phoneNumber: json['phone'],
      address: jsonAddress,
      password: json['login']['password'],
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'name': name,
      'email': email,
      'phoneNumber': phoneNumber,
      'birthDate': birthDate,
      'address': address,
      'password': password,
    };
  }
}
```

Crear Archivo en Flutter			
Acción	- Crear archivo	***Nombre***	database

Definición de parámetros a recibir o utilizar:

- String databaseName
- Database _database

Requerimiento ####

Definición de lógica a realizar

- **Base de datos SQLite:** Se debe crear un archivo que permita gestionar la base de datos local, se inicializa la base de datos y se crea la tabla que se utilizara en la aplicación.

IMAGEN 1

```
class DBSqlite {
  static const String databaseName = 'user_database.db';
  static final DBSqlite _instance = DBSqlite._internal();
  factory DBSqlite() => _instance;

  static Database? _database;

  DBSqlite._internal();

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDB();
    return _database!;
  }

  Future<Database> _initDB() async {
    String path = join(await getDatabasesPath(), databaseName);

    return await openDatabase(
      path,
      version: 1,
      onCreate: _onCreate,
    );
  }

  Future _onCreate(Database db, int version) async {
    await db.execute('''
      CREATE TABLE users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT,
        email TEXT,phoneNumber TEXT,
        birthDate TEXT,
        address TEXT,
        password TEXT
      )
    ''');
  }
```

Crear Función en Flutter			
Acción	- Crear función	***Nombre***	insertUser(User user)

Definición de parámetros a recibir o utilizar:

- User user

Definición de lógica a realizar

- **Guardar usuarios:** Se debe poder guardar los datos de los usuarios creados en la base de datos

Requerimiento ####

local (SQLite)

IMAGEN 1

```dart
Future<int> insertUser(User user) async {
  final db = await database;
  return await db.insert('users', user.toJson());
}
```

Crear Función en Flutter			
Acción	- Crear función	***Nombre***	handleRegister()

Definición de parámetros a recibir o utilizar:

- Ninguno

Definición de lógica a realizar

- **Registro de cuenta:** Al presionar el botón “Registrar Usuario” se debe guardar el usuario en la base de datos local.

IMAGEN 1

```dart
void handleRegister() async {
  User user = User(
    name: _nameController.text,
    email: _emailController.text,
    birthDate: _dateController.text,
    address: _addressController.text,
    password: _passwordController.text,
    phoneNumber: _phoneNumber.text,
  );
  database.insertUser(user);
  MessagesStatus.showStatusMessage(
      context, "Usuario registrado con exito", false);
  _nameController.clear();
  _emailController.clear();
  _dateController.clear();
  _addressController.clear();
  _phoneNumber.clear();
  _passwordController.clear();
}
```

Requerimiento ####