DÉPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Compte rendu :

State Management avec bloc

Filière :
« Ingénierie Informatique : Big Data et Cloud Computing »

II-BDCC2

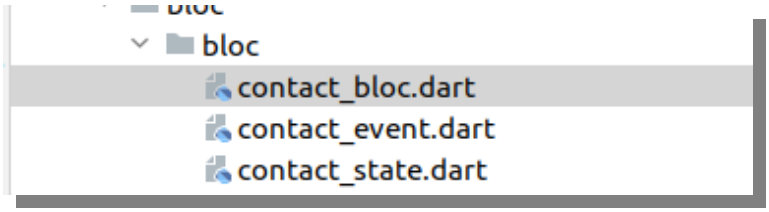
Élaboré par :

Mohamed Ait Lahcen

Lien du code source :
https://github.com/mohamed-ait/Flutter-TPs/tree/main/tp_bloc

1. Package bloc :

Ce package contient les trois classes essentielles qui permettent de gérer les éventements de l'application :



- **contact_bloc :**

  cette classe permet de répondre aux éventements de l'application sous formes des states :

```dart
class ContactBloc extends Bloc<ContactEvent,ContactState>{
  ContactRepository contactRepository;
    ContactEvent lastEvent=AucunEvenement();
  ContactBloc(ContactState contactState, this.contactRepository):super(contactState){
    on<ContactEvent>((event, emit) async{
      lastEvent=event;
     if(event is LoadAllContactsEvent){
       emit(ContactState(contacts: [],requestSate: RequestState.Loading,errorMessage: ""));
       List<Contact> contacts=await contactRepository.getAllContacts();
       emit(ContactState(contacts: contacts,requestSate: RequestState.Loaded,errorMessage: ""));
     }else  if(event is LoadBDCCContactsEvent){
       emit(ContactState(contacts: [],requestSate: RequestState.Loading,errorMessage: ""));
       try{
         List<Contact> contacts=await contactRepository.getContactsByGroup("BDCC");
         emit(ContactState(contacts: contacts,requestSate: RequestState.Loaded,errorMessage: ""));
       }catch( e){
         emit(ContactState(contacts: [],requestSate: RequestState.Error,errorMessage: e.toString()));
       }
```

- **contact_event:**

  cette classe contient l'ensemble des éventements de l'application :

```
abstract class ContactEvent{

}
class LoadAllContactsEvent extends ContactEvent{}

class LoadBDCCContactsEvent extends ContactEvent{}

class LoadGLSIDContactsEvent extends ContactEvent{}

class AucunEvenement extends ContactEvent{}
```

- **contact_state:**

  Cette qui classe qui permet au bloc de transférer les résultats :

```
1  import 'package:tp_bloc/bloc/model/ContactModel.dart';
2  enum RequestState{Loaded,Loading,Error,NONE}
3  class ContactState{
4    List<Contact> contacts =[];
5    RequestState requestSate;
6    String errorMessage;
7
8    ContactState({required this.contacts,required this.requestSate,required this.errorMessage});
9  }
```

2. Package modele :

Ce package contient la classe modele de l'application :

```
1
2  class Contact{
3    int id;
4    String name;
5    String group;
6    String profile;
7
8    Contact({required this.id, required this.name, required this.group,required this.profile});
9  }
```

3. Package repository :

Ce package contient la classe repository qui permet d'éxécuter les fonctions demandés par les éventements :

```
1
2  class Contact{
3    int id;
4    String name;
5    String group;
6    String profile;
7
8    Contact({required this.id, required this.name, required this.group,required this.profile});
9  }
```

4. Package ui.page :

ce package contient les pages de l'application voici la page contact :

```dart
class ContactPage extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Row(
        children: [
          SizedBox(width: 85,),
          Icon(Icons.contacts),
          SizedBox(width: 10,),
          Text("contacts")
        ],
      ),),  // Row, AppBar
      body: Padding(
        padding: EdgeInsets.all(10),
        child: Column(
          children: [
            BlocBuilder<ContactBloc,ContactState>(builder: (context, state) {
              // la classe qui retourne les buttons :
             return  ButtonsBar();
            }),  // BlocBuilder
```

5. Package ui.widgets :

ce package contient les différentes widgets de l'application :

- **contact_button :**

```dart
import 'package:tp_bloc/bloc/bloc/contact_event.dart';
import 'package:tp_bloc/bloc/bloc/contact_state.dart';

class ButtonContact extends StatelessWidget {
  final String text;
  final ContactEvent event;
  const ButtonContact({Key? key,required this.text,required this.event}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(onPressed: (){
      context.read<ContactBloc>().add(this.event);
      print(this.event.runtimeType.toString());
    }, child: Text(this.text,style: TextStyle(fontWeight: FontWeight.bold,fontSize: 17)),style: ElevatedButton.styleFrom(
        primary: (context.read<ContactBloc>().lastEvent.runtimeType.toString().compareTo(event.runtimeType.toString())==0 ? Colors.amberAccent :
    ));  // ElevatedButton
  }
}
```

- **button_bar :**

```dart
import 'package:tp_bloc/bloc/ui/widgets/button_contact.dart';

class ButtonsBar extends StatelessWidget {
  const ButtonsBar({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        ButtonContact(text: "All", event: LoadAllContactsEvent()),
        SizedBox(width: 5,),
        ButtonContact(text: "BDCC", event: LoadBDCCContactsEvent()),
        SizedBox(width: 5,),
        ButtonContact(text: "GLSID", event: LoadGLSIDContactsEvent()),
      ],
    );  // Row
  }
}

```

- **reload_button :**

```dart
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:tp_bloc/bloc/bloc/contact_bloc.dart';

class ReloadButton extends StatelessWidget {
  const ReloadButton({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(onPressed: (){
      ContactBloc cb=context.read<ContactBloc>();
      cb.add(cb.lastEvent);
    }, child: Icon(Icons.wifi_protected_setup_outlined),
      style: ElevatedButton.styleFrom(padding: EdgeInsets.all(3),
        fixedSize: const Size(45, 45),
        shape: const CircleBorder(),
      ),)  // ElevatedButton
    ;
  }
}

```

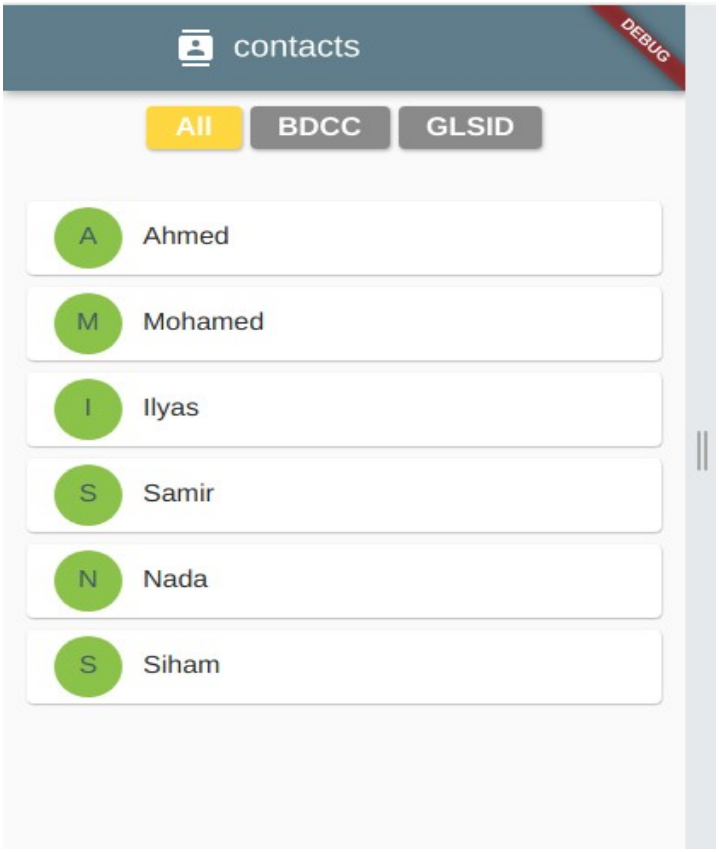
- **contacts_list :**

```dart
import 'package:tp_bloc/bloc/bloc/contact_state.dart';

class ListContact extends StatelessWidget {
  final ContactState state;
  const ListContact({Key? key,required this.state}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return  ListView.builder(
      itemCount: state.contacts.length,
      itemBuilder: (context,index){
        return Card(
          child: ListTile(
            leading: CircleAvatar(backgroundColor: Colors.lightGreen,child: Text("${state.contacts[index].prof
            title: Text(state.contacts[index].name),

          ),  // ListTile
        );  // Card

      },
    );  // ListView.builder
  }
}
```

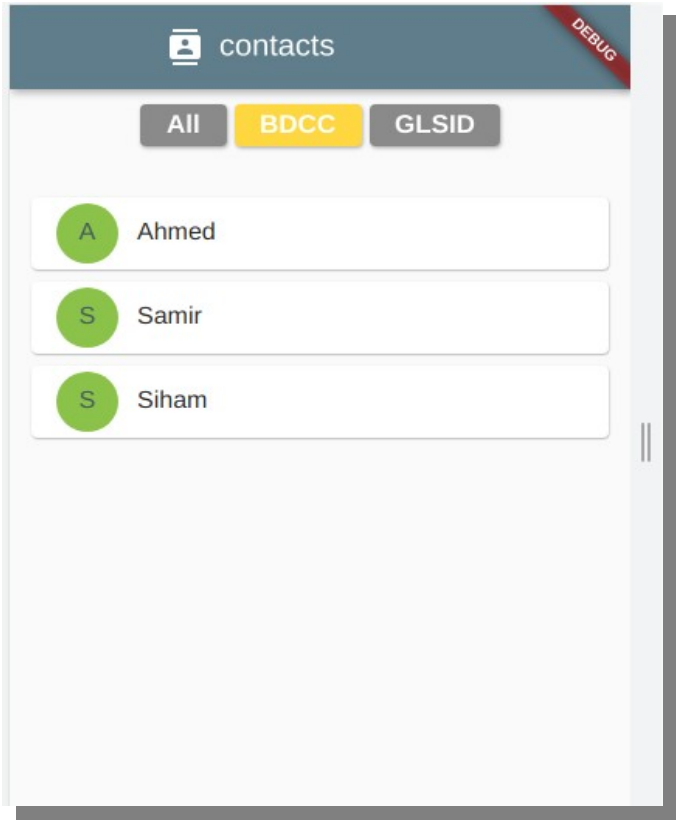
6. La représentation graphique de l'application :

a- La page de l'application avant d'envoyer aucun événement :

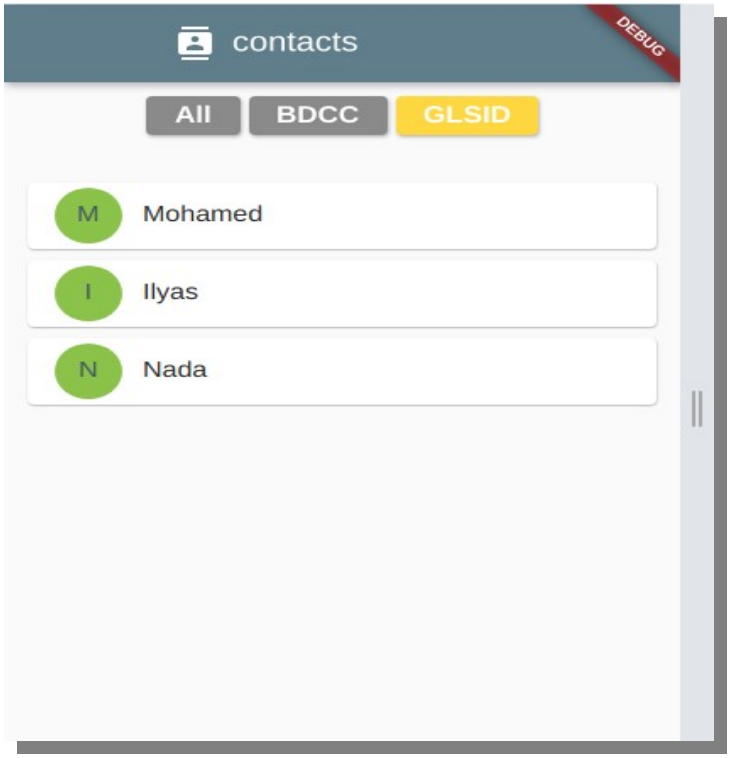
b- La listes de tous les contacts :



c- La listes des contacts du groupe BDCC :



d- La listes des contacts GLSID :



e- La page dans le cas d'error :