Документація по застосунку “Plan Everyday”

**Назва застосунку**: Plan EveryDay

**Опис**: Plan Everyday – крос-платформний застосунок для створення, редагування та зберігання записів трьох типів: запис (стандартний запис для зберігання якоїсь інформації), подія і завдання.

**Будова застосунку (вікна)**:

* Home Screen – головне і початкове вікно застосунку з верхнім меню, кнопкою створення нового запису, та кнопкою створення нового запису.
* Create Note – вікно створення нового запису.
* Edit Note – вікно для редагування створених записів.
* Search Screen – вікно пошуку створених записів за назвою або описом.

**Використані технології**: фреймворк Flutter для створення вигляду і функцій застосунку, локальна база даних Hive для легкого збереження ключів і значень, мова програмування Dart, бібліотека Cubit для керування станом екрана та реалізації шаблону проектування BLoC, фреймворк Riverpod для легшої роботи з асинхронним кодом.

**Функції**:

* Створення нового запису.

Опис: Створюваний запис має наступний перелік параметрів: назва (Title), опис (Description), тип (Type), дата (Data), час (Time). Назва та опис вводяться в CreateNoteField, який приймає їх та зберігає як типи даних String. Тип вибирається за допомогою радіо-кнопок в класі ChooseTypeWidget, індекс вибраного типу зберігаються в BlocProvider, а вже за цим індексом визначається назва вибраного типу, яка також зберігається під типом String. Схожим чином вибираються та зберігаються дата та час.

Код області вводу назви та опису:

class CreateNoteField extends StatelessWidget {

  const CreateNoteField(

    {super.key, required this.hint, this.maxLines = 1, this.onSaved, this.onChanged});

  final String hint;

  final int maxLines;

  final void Function(String?)? onSaved;

  final Function(String)? onChanged;

  @override

  Widget build(BuildContext context) {

    return TextFormField(

      onChanged: onChanged,

      onSaved: onSaved,

      validator: (value){

        if(value?.isEmpty ?? true) {

          return 'Field is required';

        }

        else {

          return null;

        }

      },

      // Стиль тексту у віджеті

      style: TextStyle(color: Colors.white),

      // Колір курсора

      cursorColor: Color.fromARGB(255, 32, 166, 255),

      maxLines: maxLines,

      decoration: InputDecoration(

        hintText: hint,

        hintStyle: TextStyle(

          color: Color.fromARGB(255, 56, 80, 156),

          fontWeight: FontWeight.w400,

        ),

        border: buildBorder(),

        enabledBorder: buildBorder(),

        focusedBorder: buildBorder(Color.fromARGB(255, 32, 166, 255)),

      ),

    );

  }

  // Створення границі текстового поля

  OutlineInputBorder buildBorder([color]) {

    return OutlineInputBorder(

      borderRadius: BorderRadius.circular(8),

      borderSide: BorderSide(

        color: color ?? Colors.white,

      )

    );

  }

}

Код радіо-кнопок для вибору типу:

class RadioWidget extends StatelessWidget {

  final Color typeColor;

  final String titleRadio;

  final int valueInput;

  final Function onChangeValue;

  const RadioWidget({

    Key? key,

    required this.typeColor,

    required this.titleRadio,

    required this.valueInput,

    required this.onChangeValue,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return BlocBuilder<TypeCubit, int>(

      builder: (context, selectedType) {

        return RadioListTile<int>(

          value: valueInput,

          groupValue: selectedType,

          onChanged: (value) => onChangeValue(),

          title: Text(

            titleRadio,

            style: TextStyle(

              color: typeColor,

              fontWeight: FontWeight.bold,

            ),

          ),

          activeColor: typeColor,

        );

      },

    );

  }

}

Код вибору дати та часу:

class DateTimeWidget extends StatelessWidget {

  const DateTimeWidget({

    super.key,

    required this.titleText,

    required this.valueText,

    required this.iconSection,

    required this.onTap,

  });

  final String titleText;

  final String valueText;

  final IconData iconSection;

  final VoidCallback onTap;

  @override

  Widget build(BuildContext context) {

    return Expanded(

      child: Column(

        crossAxisAlignment: CrossAxisAlignment.start,

        children: [

          Text(

            titleText,

            style: TextStyle(

              fontSize: 16,

              fontWeight: FontWeight.w500,

              color: Colors.white,

            ),

          ),

          const SizedBox(height: 4),

          Material(

            borderRadius: BorderRadius.circular(10.5),

            clipBehavior: Clip.antiAlias,

            child: Ink(

              decoration: BoxDecoration(

                color: Color.fromARGB(255, 45, 62, 118),

                borderRadius: BorderRadius.circular(10),

              ),

              child: InkWell(

                onTap: onTap,

                borderRadius: BorderRadius.circular(10),

                child: Container(

                  padding: const EdgeInsets.symmetric(

                    horizontal: 12,

                    vertical: 12,

                  ),

                  decoration: BoxDecoration(

                    color: Colors.transparent,

                    borderRadius: BorderRadius.circular(10),

                  ),

                  child: Row(

                    children: [

                      Icon(iconSection, color: Colors.white),

                      const SizedBox(width: 8),

                      Text(

                        valueText,

                        style: TextStyle(color: Color.fromRGBO(238, 238, 238, 1)),

                      ),

                    ],

                  ),

                ),

              ),

            ),

          ),

        ],

      ),

    );

  }

}

* Зберігання записів.

Опис: Зберігання створених записів відбувається за допомогою локальної бази даних Hive. Усі параметри зберігаються в типах даних String з присвоїними ним індексами у віджеті, що відповідає за модель запису.

Код:

import 'package:hive/hive.dart';

part 'note\_model.g.dart';

@HiveType(typeId: 0)

class NoteModel extends HiveObject {

  @HiveField(0)

  String title;

  @HiveField(1)

  String description;

  @HiveField(2)

  String type;

  @HiveField(3)

  String dateValue;

  @HiveField(4)

  String timeValue;

  NoteModel({

    required this.title,

    required this.description,

    required this.type,

    required this.dateValue,

    required this.timeValue

  });

}

* Редагування записів.

Опис: відбувається за тим же принципом, що і створення запису, але з можливість змінити вже введенні назву та опис. Функцію редагування (внесення подальших змін) можна побачити знову ж таки в коді класі CreateNoteField.

* Пошук записів.

Опис: пошук створених записів відбувається шляхом фільтрації всього списку через введений у верхню область текст. Зшукаються зпівпадіння між цим текстом, та текстовими значеннями параметрів title та description. При не знаходженні зпівпадінь виводиться відповідне повідомлення.

Код:

// Вікно для пошуку заміток

class SearchScreen extends StatefulWidget {

  @override

  \_SearchScreenState createState() => \_SearchScreenState();

}

class \_SearchScreenState extends State<SearchScreen> {

  // Змінна для збереження введеного пошукового запиту

  String query = '';

  @override

  void initState() {

    super.initState();

    // Завантаження всіх заміток при ініціалізації екрану

    BlocProvider.of<NotesCubit>(context).fetchAllNotes();

  }

  // Метод для пошуку заміток за введеним запитом

  void searchNotes(String query) {

    final notesCubit = BlocProvider.of<NotesCubit>(context);

    notesCubit.searchNotes(query);

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: const Color.fromRGBO(151, 187, 255, 1),

        // Поле для введення пошукового запиту

        title: TextField(

          onChanged: (value) {

            setState(() {

              query = value;

            });

            // Виклик методу пошуку при зміні тексту в полі

            searchNotes(value);

          },

          decoration: InputDecoration(

            hintText: 'Search...',

            border: InputBorder.none,

          ),

          style: TextStyle(color: Colors.white, fontSize: 18),

        ),

      ),

      backgroundColor: const Color.fromRGBO(27, 29, 84, 1),

      // Віджет для побудови UI в залежності від стану NotesCubit

      body: BlocBuilder<NotesCubit, NotesState>(

        builder: (context, state) {

          if (state is NotesLoading) {

            // Показати індикатор завантаження, коли замітки завантажуються

            return Center(child: CircularProgressIndicator());

          } else if (state is NotesLoaded) {

            List<NoteModel> notes = state.notes;

            // Показати список заміток, коли вони завантажені

            if (notes.isEmpty) {

              return Center(child: Text("No notes found"));

            }

            return ListView.builder(

              itemCount: notes.length,

              itemBuilder: (context, index) {

                return ListTile(

                  title: Text(notes[index].title, style: TextStyle(color: Colors.white)),

                  subtitle: Text(notes[index].description, style: TextStyle(color: Colors.white70)),

                );

              },

            );

          } else if (state is NotesError) {

            return Center(child: Text('Error: ${state.error}', style: TextStyle(color: Colors.red)));

          } else {

            return Container();

          }

        },

      ),

    );

  }

}

*Додатково*: До даного застосунку не має спроможності створити UML документацію, через не сумісність в одній програмі версій бібліотек Hive та DCDG.