МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

УНИВЕРСИТЕТ САТПАЕВ



ЛAБОРАТОРНЫЙ ПРОЕКТ

Тема: Мобильное приложение Календарь событий

|  |  |  |  |
| --- | --- | --- | --- |
| № | Качество выполнения работы | Диапазон  оценки | Получено,  % |
| 1 | Не выполнено | 0% |  |
| 2 | Выполнено | 0-50% |  |
| 3 | Самостоятельная систематизация материала | 0-10% |  |
| 4 | Выполнение требуемого объема и в указанный срок | 0-5% |  |
| 5 | Использование дополнительной научной литературы | 0-5% |  |
| 6 | Уникальность выполненного задания | 0-10% |  |
| 7 | Защита работы | 0-20% |  |
|  | Итого: | 0-100% |  |

Преподаватель: Аубакиров Бакдаулет

Студент: Токтархан Алишер

Специальность: Computer Science

Алматы 2023 г

**Содержание**

1.Введение……………………………………………………………….………………3

2.О проекте………………………………………………………………………………3

3.Вывод…………………………………………………………………………………….9

4.Код…………………………………………………………………………………………10

**1.Введение**

Это приложение предназначено для управления вашими ежедневными делами и позволяет планировать встречи, важные события и напоминания. С помощью этого приложения вы сможете легко и удобно организовать свою жизнь и не пропустить ни одно важное событие. В этой документации мы расскажем, как использовать приложение и какие функции в нем доступны. Мы постарались сделать ее максимально понятной и информативной, чтобы вы смогли быстро начать работу с приложением и использовать все его возможности. Приложение написано на Flutter.

Flutter — комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, веб-приложений, а также настольных приложений под Windows, macOS и Linux с использованием языка программирования Dart, разработанный и развиваемый корпорацией Google.

**2.О проекте**

Приложение “**Календарь с событиями**” на Flutter предоставляет удобный и интуитивно понятный интерфейс для планирования задач, хранения заметок и отслеживания своего расписания. Календарь отображает один экран в котором календарь в формате месяца. Календарь можно перелистывать горизонтально. В верхней панели отображается месяц и год на котором пользователь находится а также две кнопки удаления и перехода к текущему месяцу. Также есть кнопка добавления события, события могут быть созданы для одного дня или для определенного периода времени. При нажатии кнопки добавления события всплывает окошко добавления события, в окошке можно написать название и выбрать цвет и период времени события.

Для просмотра информации события необходимо нажать на ячейку дня для просмотря всплывает лист событий, он реагирует на жесты перетаскивания.

Приложение использует материальный дизайн и имеет темную цветовую схему, которая была настроена в ThemeData в файле main.dart. Также использует несколько пакетов, таких как cr\_calendar для реализации календаря и событий intl для форматирования даты.

**Главный экран**

Класс **CalendarPage** представляет главную страницу приложения, на которой расположен календарь и управляющие элементы.

Календарь реализован с использованием библиотеки cr\_calendar. Он отображает дни месяца, разделяет их по неделям и позволяет пользователю просматривать события, связанные с конкретными днями.

Класс \_**CalendarPageState** содержит логику управления календарем и обработки событий пользователя. В методе initState происходит инициализация календаря и добавление примеров событий. В методе dispose происходит очистка ресурсов, занятых календарем.

Окно для создания события сделано с помощью виджета CreateEventDialog.

showModalBottomSheet - Показывает модальный нижний лист Material Design. Параметр isScrollControlledуказывает, является ли это маршрутом для нижнего листа, который будет использовать DraggableScrollableSheet .

Основные методы класса \_CalendarPageState включают:

**\_changeCalendarPage** - переход на следующий или предыдущий месяц календаря

\_**onCalendarPageChanged** - обработка смены месяца календаря \_setTexts установка текста заголовка на основе выбранного месяца и года

**\_showCurrentMonth** - переход к текущему месяцу

\_**addEvent** - открытие диалогового окна для создания нового события

\_**createExampleEvents** - добавление примеров событий в календарь

\_**showDayEventsInModalSheet** - отображение модального окна с событиями конкретного дня календаря.

Кроме того, класс \_**CalendarPageState** определяет виджеты, используемые для отображения элементов календаря, такие как **DayItemWidget**, **WeekDaysWidget** и **EventWidget**.

**DayItemWidget -** Виджет ячейки дня для календаря. DayItemWidget принимает в качестве входных данных объект DayItemProperties, который содержит информацию об отображаемом дне. Затем виджет использует эту информацию для определения внешнего вида ячейки.

**WeekDaysWidget** - Виджет, представляющий дни недели в строке над представлением календарного месяца. Виджет возвращает контейнер высотой 40 и центрированный текстовый виджет внутри него. Виджет «Текст» отображает первую букву названия дня в верхнем регистре (например, «М» для понедельника) и окрашен в фиолетовый цвет, непрозрачный на 90 %.

**EventWidget** - Пользовательский виджет событий с закругленными границами. Виджет принимает в своем конструкторе объект EventProperties, который содержит информацию о событии, такую как его имя и цвет фона.

В целом, этот код позволяет создавать календарное приложение с использованием Flutter и библиотеки cr\_calendar.

**Изображение выглядит как текст, в помещении, электроника, черный

Автоматически созданное описание**

Рисунок – 1 Главный экран

**Всплывающий диалог для создания события.**

**CreateEventDialog**

Диалоговое окно позволяет пользователям вводить название события, выбирать цвет для события и выбирать диапазон дат для события с помощью средства выбора даты.

Функция \_**selectColor** вызывается при выборе цвета и обновляет индекс выбранного цвета.

Функция \_**setRangeData** вызывается при выборе диапазона дат и обновляет переменные диапазона дат и текст кнопки выбора диапазона дат. Функция \_parseDateRange используется для форматирования выбранного диапазона дат в удобочитаемый формат.

Функция \_**showRangePicker** показывает календарь во всплывающем диалоговом окне для выбора диапазона дат для события календаря. В функции используется виджет **PickerDayItemWidget**.

Виджет **PickerDayItemWidget** принимает объект DayItemProperties, который содержит информацию об отображаемом дне (например, номер дня, в текущем месяце или нет и т. д.). Виджет PickerDayItemWidget отображает круглый номер дня в центре ячейки. Если день находится в пределах выбранного диапазона дат, фон окрашивается в полупрозрачный фиолетовый цвет.

Для отображения название события используется виджет **date\_picker\_title\_widget.dart** в котором метод build создает контейнер, в котором отображается текст созданной события.

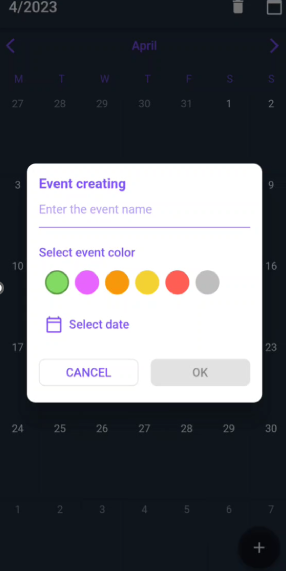


Рисунок – 2 Окошко для создания события

**Лист событий в выбранный день**

**DayEventsBottomSheet** class Перетаскиваемый нижний лист с событиями дня. Он принимает три обязательных параметра:

events — список объектов CalendarEventModel.

day — объект DateTime, представляющий день, для которого отображаются события.

screenHeight — высота экрана.

Виджет использует DraggableScrollableSheet в качестве основного компонента, который позволяет пользователю перетаскивать нижний лист вверх или вниз, чтобы показать больше или меньше событий. Если список событий пуст, виджет отображает сообщение о том, что в выбранный день нет событий. В противном случае он показывает ListView.builder, который отображает каждое событие в виде виджета Card с цветной полосой слева, указывающей тип события.

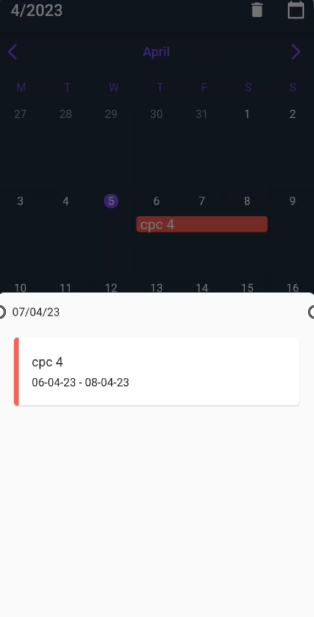


Рисунок-3 Лист событий в выбранном дне, если в выбранном дне есть событие

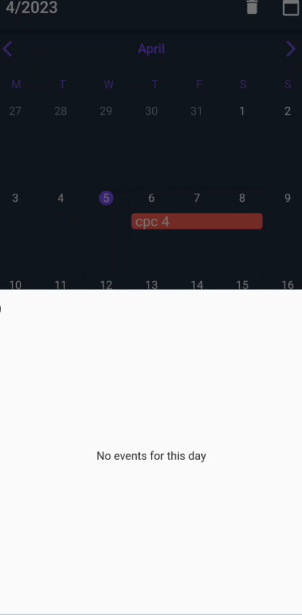


Рисунок-4 Лист событий в выбранном дне, если в выбранном дне нету событий

Также в проекте используются утилиты “constants”, “extensions”

Constants хранит формат даты.

Extensions хранит метод для форматирования.

В colors.dart хранятся данные о цветах для кастомизации виджетов.

В lib находися файл main.dart и директории widgets, utils, res, pages.

**3.Вывод**

Приложение календаря с событиями предоставляет удобный и интуитивно понятный интерфейс для планирования задач, хранения заметок и отслеживания своего расписания.

**4.Код**

**lib/main.dart:**

import 'package:cr\_calendar\_example/pages/calendar\_page.dart';

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:flutter/material.dart';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatefulWidget {

  const MyApp({super.key});

  @override

  \_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      /// Theme.

      theme: ThemeData(

        floatingActionButtonTheme:

            const FloatingActionButtonThemeData(backgroundColor: Colors.black38),

        primaryColor: Colors.black38,

        colorScheme: ColorScheme.fromSwatch().copyWith(secondary: violet),

        iconTheme: const IconThemeData(color: violet),

        outlinedButtonTheme: OutlinedButtonThemeData(

          style: OutlinedButton.styleFrom(

            textStyle: const TextStyle(

              color: Colors.black38,

              fontSize: 16,

              fontWeight: FontWeight.w500,

            ),

            foregroundColor: violet,

            shape: const RoundedRectangleBorder(

                borderRadius: BorderRadius.all(Radius.circular(8))),

          ),

        ),

        elevatedButtonTheme: ElevatedButtonThemeData(

          style: ElevatedButton.styleFrom(

            textStyle: const TextStyle(

              fontSize: 16,

              fontWeight: FontWeight.w500,

            ),

            shadowColor: Colors.transparent,

            elevation: 0,

            foregroundColor: violet,

            shape: const RoundedRectangleBorder(

                borderRadius: BorderRadius.all(Radius.circular(8))),

          ),

        ),

        dialogTheme: const DialogTheme(

          shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.all(Radius.circular(12))),

        ),

        appBarTheme: const AppBarTheme(

          color: Color(0xFF212F3D)

        ),

      ),

      home: const CalendarPage(),

    );

  }

}

**Lib/widgets/event\_widget.dart:**

import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:flutter/material.dart';

/// Пользовательский виджет событий с закругленными границами

///

/// Виджет принимает в своем конструкторе объект EventProperties,

/// который содержит информацию о событии, такую как его имя и цвет фона.

class EventWidget extends StatelessWidget {

  const EventWidget({

    required this.drawer,

    super.key,

  });

  final EventProperties drawer;

  @override

  Widget build(BuildContext context) {

    return Container(

      margin: const EdgeInsets.symmetric(horizontal: 3),

      padding: const EdgeInsets.symmetric(horizontal: 5),

      decoration: BoxDecoration(

        borderRadius: const BorderRadius.all(Radius.circular(4)),

        color: drawer.backgroundColor,

      ),

      child: FittedBox(

        fit: BoxFit.fitHeight,

        alignment: Alignment.centerLeft,

        child: Text(

          drawer.name,

          overflow: TextOverflow.ellipsis,

          style: const TextStyle(color: Colors.white),

        ),

      ),

    );

  }

}

**Lib/widgets/week\_days\_widget.dart:**

import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';

/// Виджет, представляющий дни недели в строке над представлением календарного месяца.

///

/// Виджет возвращает контейнер высотой 40 и центрированный текстовый виджет внутри него.

/// Виджет «Текст» отображает первую букву названия дня в верхнем регистре (например, «М» для понедельника)

/// и окрашен в фиолетовый цвет, непрозрачный на 90 %.

class WeekDaysWidget extends StatelessWidget {

  const WeekDaysWidget({

    required this.day,

    super.key,

  });

  /// Значение [WeekDay] из [WeekDaysBuilder].

  final WeekDay day;

  @override

  Widget build(BuildContext context) {

    return Container(

      height: 40,

      child: Center(

        child: Text(

          describeEnum(day).substring(0, 1).toUpperCase(),

          style: TextStyle(

            color: violet.withOpacity(0.9),

          ),

        ),

      ),

    );

  }

}

**Lib/widgets/day\_item \_widget.dart:**

import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:flutter/material.dart';

/// Виджет ячейки дня для календаря

/// Это виджет Flutter, который отображает ячейку одного дня в календаре.

///

/// DayItemWidget принимает в качестве входных данных объект DayItemProperties, который содержит информацию об отображаемом дне.

/// Затем виджет использует эту информацию для определения внешнего вида ячейки.

///

/// Оператор if внизу условно отображает индикатор количества событий в зависимости от того,

/// есть ли какие-либо события, которые не поместились в ячейку.

class DayItemWidget extends StatelessWidget {

  const DayItemWidget({

    required this.properties,

    super.key,

  });

  final DayItemProperties properties;

  @override

  Widget build(BuildContext context) {

    return Container(

      decoration: BoxDecoration(

          border: Border.all(color: Color(0xFF1C2833).withOpacity(1), width: 0.3)),

      child: Stack(

        children: [

          Container(

            padding: const EdgeInsets.only(top: 4),

            alignment: Alignment.topCenter,

            child: Container(

              height: 18,

              width: 18,

              decoration: BoxDecoration(

                color: properties.isCurrentDay ? violet : Colors.transparent,

                shape: BoxShape.circle,

              ),

              child: Center(

                child: Text('${properties.dayNumber}',

                    style: TextStyle(

                        color: properties.isCurrentDay

                            ? Colors.white

                            : Colors.white

                                .withOpacity(properties.isInMonth ? 1 : 0.5))),

              ),

            ),

          ),

          if (properties.notFittedEventsCount > 0)

            Container(

              padding: const EdgeInsets.only(right: 2, top: 2),

              alignment: Alignment.topRight,

              child: Text('+${properties.notFittedEventsCount}',

                  style: TextStyle(

                      fontSize: 10,

                      color:

                          Colors.white.withOpacity(properties.isInMonth ? 1 : 0.5))),

            ),

        ],

      ),

    );

  }

}

**Lib/widgets/create\_event\_dialog.dart:**

import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:cr\_calendar\_example/utills/constants.dart';

import 'package:cr\_calendar\_example/utills/extensions.dart';

import 'package:cr\_calendar\_example/widgets/date\_picker\_title\_widget.dart';

import 'package:cr\_calendar\_example/widgets/picker\_day\_item\_widget.dart';

import 'package:cr\_calendar\_example/widgets/week\_days\_widget.dart';

import 'package:flutter/material.dart';

import 'package:intl/intl.dart';

/// Всплывающий диалог для создания события.

///

///Диалоговое окно позволяет пользователям вводить название события,

///выбирать цвет для события и выбирать диапазон дат для события с помощью средства выбора даты.

///

/// Функция \_selectColor вызывается при выборе цвета и обновляет индекс выбранного цвета.

///

/// Функция \_setRangeData вызывается при выборе диапазона дат и обновляет переменные диапазона дат

/// и текст кнопки выбора диапазона дат.

///

/// Функция \_parseDateRange используется для форматирования выбранного диапазона дат в удобочитаемый формат.

class CreateEventDialog extends StatefulWidget {

  const CreateEventDialog({super.key});

  @override

  \_CreateEventDialogState createState() => \_CreateEventDialogState();

}

class \_CreateEventDialogState extends State<CreateEventDialog> {

  int \_selectedColorIndex = 0;

  final \_eventNameController = TextEditingController();

  String \_rangeButtonText = 'Select date';

  DateTime? \_beginDate;

  DateTime? \_endDate;

  @override

  void dispose() {

    \_eventNameController.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    final size = MediaQuery.of(context).size;

    return Dialog(

      child: ConstrainedBox(

        constraints: BoxConstraints(

          maxHeight: size.height \* 0.7,

          maxWidth: size.width \* 0.8,

        ),

        child: SingleChildScrollView(

          child: Padding(

            padding: const EdgeInsets.all(16),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              mainAxisSize: MainAxisSize.min,

              children: [

                /// Название диалога.

                const Text(

                  'Event creating',

                  style: TextStyle(

                    color: violet,

                    fontWeight: FontWeight.bold,

                    fontSize: 18,

                  ),

                ),

                /// Поле ввода названия события.

                TextField(

                  cursorColor: violet,

                  style: const TextStyle(color: violet, fontSize: 16),

                  decoration: InputDecoration(

                    enabledBorder: UnderlineInputBorder(

                      borderSide: BorderSide(color: violet.withOpacity(1)),

                    ),

                    hintText: 'Enter the event name',

                    hintStyle:

                        TextStyle(color: violet.withOpacity(0.6), fontSize: 16),

                  ),

                  controller: \_eventNameController,

                ),

                const SizedBox(height: 24),

                /// Раздел выбора цвета.

                const Text(

                  'Select event color',

                  style: TextStyle(

                    fontSize: 16,

                    color: violet,

                    fontWeight: FontWeight.w500,

                  ),

                ),

                const SizedBox(height: 14),

                /// Строка выбора цвета.

                SingleChildScrollView(

                  scrollDirection: Axis.horizontal,

                  child: Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: [

                      ...List.generate(

                        eventColors.length,

                        (index) => GestureDetector(

                          onTap: () {

                            \_selectColor(index);

                          },

                          child: Padding(

                            padding: const EdgeInsets.only(left: 8),

                            child: Container(

                              foregroundDecoration: BoxDecoration(

                                border: index == \_selectedColorIndex

                                    ? Border.all(

                                        color: Colors.black.withOpacity(0.3),

                                        width: 2)

                                    : null,

                                shape: BoxShape.circle,

                                color: eventColors[index],

                              ),

                              width: 32,

                              height: 32,

                            ),

                          ),

                        ),

                      )

                    ],

                  ),

                ),

                const SizedBox(height: 16),

                /// Кнопка выбора даты.

                TextButton(

                  onPressed: \_showRangePicker,

                  child: Row(

                    children: [

                      const Icon(

                        Icons.calendar\_today\_outlined,

                        color: violet,

                      ),

                      const SizedBox(width: 8),

                      Text(

                        \_rangeButtonText,

                        style: const TextStyle(

                          fontSize: 16,

                          color: violet,

                          fontWeight: FontWeight.w500,

                        ),

                      ),

                    ],

                  ),

                ),

                const SizedBox(height: 16),

                Row(

                  mainAxisAlignment: MainAxisAlignment.spaceAround,

                  children: [

                    /// Кнопка отмены.

                    Expanded(

                      child: OutlinedButton(

                        onPressed: () {

                          Navigator.of(context).pop();

                        },

                        child: const Text('CANCEL'),

                      ),

                    ),

                    const SizedBox(width: 16),

                    /// кнопка ОК.

                    Expanded(

                      child: ElevatedButton(

                        onPressed:

                            \_validateEventData() ? \_onEventCreation : null,

                        child: const Text('OK'),

                      ),

                    ),

                  ],

                ),

              ],

            ),

          ),

        ),

      ),

    );

  }

  /// Выберите цвет одним нажатием.

  void \_selectColor(int index) {

    setState(() {

      \_selectedColorIndex = index;

    });

  }

  /// Установить текст кнопки выбора диапазона.

  void \_setRangeData(DateTime? begin, DateTime? end) {

    if (begin == null || end == null) {

      return;

    }

    setState(() {

      \_beginDate = begin;

      \_endDate = end;

      \_rangeButtonText = \_parseDateRange(begin, end);

    });

  }

  /// Разобрать выбранную дату в читаемый формат.

  String \_parseDateRange(DateTime begin, DateTime end) {

    if (begin.isAtSameMomentAs(end)) {

      return begin.format(kDateRangeFormat);

    } else {

      return '${begin.format(kDateRangeFormat)} - ${end.format(kDateRangeFormat)}';

    }

  }

  /// Проверить информацию о событии для включения кнопки «ОК».

  bool \_validateEventData() {

    return \_eventNameController.text.isNotEmpty &&

        \_beginDate != null &&

        \_endDate != null;

  }

  /// Закройте диалог и передайте [CalendarEventModel] в качестве аргумента.

  void \_onEventCreation() {

    final beginDate = \_beginDate;

    final endDate = \_endDate;

    if (beginDate == null || endDate == null) {

      return;

    }

    Navigator.of(context).pop(

      CalendarEventModel(

        name: \_eventNameController.text,

        begin: beginDate,

        end: endDate,

        eventColor: eventColors[\_selectedColorIndex],

      ),

    );

  }

  /// Показать календарь во всплывающем диалоговом окне для выбора диапазона дат для события календаря.

  void \_showRangePicker() {

    FocusScope.of(context).unfocus();

    showCrDatePicker(

      context,

      properties: DatePickerProperties(

        onDateRangeSelected: \_setRangeData,

        dayItemBuilder: (properties) =>

            PickerDayItemWidget(properties: properties),

        weekDaysBuilder: (day) => WeekDaysWidget(day: day),

        initialPickerDate: \_beginDate ?? DateTime.now(),

        pickerTitleBuilder: (date) => DatePickerTitle(date: date),

        yearPickerItemBuilder: (year, isPicked) => Container(

          height: 24,

          width: 54,

          decoration: BoxDecoration(

            color: isPicked ? violet : Colors.white,

            borderRadius: const BorderRadius.all(Radius.circular(8)),

          ),

          child: Center(

            child: Text(year.toString(),

                style: TextStyle(

                    color: isPicked ? Colors.white : violet, fontSize: 16)),

          ),

        ),

        controlBarTitleBuilder: (date) => Text(

          DateFormat(kAppBarDateFormat).format(date),

          style: const TextStyle(

            fontSize: 16,

            color: violet,

            fontWeight: FontWeight.normal,

          ),

        ),

        okButtonBuilder: (onPress) => ElevatedButton(

          onPressed: () => onPress?.call(),

          child: const Text('OK'),

        ),

        cancelButtonBuilder: (onPress) => OutlinedButton(

          onPressed: () => onPress?.call(),

          child: const Text('CANCEL'),

        ),

      ),

    );

  }

}

**Lib/widgets/date\_picker\_title\_widget.dart:**

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:cr\_calendar\_example/utills/constants.dart';

import 'package:cr\_calendar\_example/utills/extensions.dart';

import 'package:flutter/material.dart';

/// В методе build создается контейнер, в котором отображается текст созданной события.

/// Константа kMonthFormatWidthYear, которая определяет формат даты (например, "April 2023").

///

/// Затем этот текст отображается в виде текстового виджета с помощью Text,

/// настраиваемого с использованием нескольких свойств стиля (размер шрифта, цвет и жирность).

class DatePickerTitle extends StatelessWidget {

  const DatePickerTitle({

    required this.date,

    super.key,

  });

  final DateTime date;

  @override

  Widget build(BuildContext context) {

    return Container(

        padding: const EdgeInsets.only(top: 16),

        alignment: Alignment.centerLeft,

        child: Text(

          date.format(kMonthFormatWidthYear),

          style: const TextStyle(

            fontSize: 21,

            color: violet,

            fontWeight: FontWeight.w500,

          ),

        ));

  }

}

**Lib/widgets/picker\_day\_item\_widget.dart:**

import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:flutter/material.dart';

/// Виджет ячейки дня для выбора диапазона.

///

/// Виджет PickerDayItemWidget принимает объект DayItemProperties,

/// который содержит информацию об отображаемом дне (например, номер дня, в текущем месяце или нет и т. д.).

///

/// Виджет PickerDayItemWidget отображает круглый номер дня в центре ячейки. Если день находится в пределах выбранного диапазона дат,

/// фон окрашивается в полупрозрачный фиолетовый цвет.

class PickerDayItemWidget extends StatelessWidget {

  const PickerDayItemWidget({

    required this.properties,

    super.key,

  });

  final DayItemProperties properties;

  @override

  Widget build(BuildContext context) {

    /// Блокировать соотношение сторон элементов, чтобы они были прямоугольниками.

    return AspectRatio(

      aspectRatio: 1 / 1,

      child: Stack(

        children: [

          /// Полупрозрачный фиолетовый фон для дней в выбранном диапазоне.

          if (properties.isInRange)

            /// Для первых и последних дней в диапазоне цвет фона виден только с одной стороны.

            Row(

              children: [

                Expanded(

                    child: Container(

                        color: properties.isFirstInRange

                            ? Colors.transparent

                            : violet.withOpacity(0.4))),

                Expanded(

                    child: Container(

                        color: properties.isLastInRange

                            ? Colors.transparent

                            : violet.withOpacity(0.4))),

              ],

            ),

          Container(

            decoration: BoxDecoration(

              shape: BoxShape.circle,

              color: properties.isFirstInRange ||

                      properties.isLastInRange ||

                      properties.isSelected

                  ? violet

                  : Colors.transparent,

            ),

            child: Center(

              child: Text('${properties.dayNumber}',

                  style: TextStyle(

                      color: properties.isInRange || properties.isSelected

                          ? Colors.white

                          : violet

                              .withOpacity(properties.isInMonth ? 1 : 0.5))),

            ),

          ),

        ],

      ),

    );

  }

}

**Lib/widgets/** **day\_events\_bottom\_sheet.dart:**

import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:cr\_calendar\_example/utills/constants.dart';

import 'package:cr\_calendar\_example/utills/extensions.dart';

import 'package:flutter/material.dart';

/// Перетаскиваемый нижний лист с событиями дня.

///

/// Он принимает три обязательных параметра: events — список объектов CalendarEventModel,

/// day — объект DateTime, представляющий день, для которого отображаются события, и screenHeight — высота экрана.

///

/// Виджет использует DraggableScrollableSheet в качестве основного компонента,

/// который позволяет пользователю перетаскивать нижний лист вверх или вниз, чтобы показать больше или меньше событий.

///

/// Если список событий пуст, виджет отображает сообщение о том, что в выбранный день нет событий.

/// В противном случае он показывает ListView.builder,

/// который отображает каждое событие в виде виджета Card с цветной полосой слева, указывающей тип события.

class DayEventsBottomSheet extends StatelessWidget {

  const DayEventsBottomSheet({

    required this.screenHeight,

    required this.events,

    required this.day,

    super.key,

  });

  final List<CalendarEventModel> events;

  final DateTime day;

  final double screenHeight;

  @override

  Widget build(BuildContext context) {

    return DraggableScrollableSheet(

        maxChildSize: 0.9,

        expand: false,

        builder: (context, controller) {

          return events.isEmpty

              ? const Center(child: Text('No events for this day'))

              : ListView.builder(

                  controller: controller,

                  itemCount: events.length + 1,

                  itemBuilder: (context, index) {

                    if (index == 0) {

                      return Padding(

                        padding: const EdgeInsets.only(

                          left: 18,

                          top: 16,

                          bottom: 16,

                        ),

                        child: Text(day.format('dd/MM/yy')),

                      );

                    } else {

                      final event = events[index - 1];

                      return Container(

                          height: 100,

                          child: Padding(

                              padding: const EdgeInsets.symmetric(

                                  horizontal: 16, vertical: 4),

                              child: Card(

                                  clipBehavior: Clip.antiAlias,

                                  child: Row(

                                    children: [

                                      Container(

                                        color: event.eventColor,

                                        width: 6,

                                      ),

                                      Expanded(

                                          child: Padding(

                                        padding:

                                            const EdgeInsets.only(left: 16),

                                        child: Align(

                                          alignment: Alignment.centerLeft,

                                          child: Column(

                                            mainAxisAlignment:

                                                MainAxisAlignment.center,

                                            crossAxisAlignment:

                                                CrossAxisAlignment.start,

                                            children: [

                                              Text(

                                                event.name,

                                                style: const TextStyle(

                                                    fontSize: 16),

                                              ),

                                              const SizedBox(height: 8),

                                              Text(

                                                '${event.begin.format(kDateRangeFormat)} - '

                                                '${event.end.format(kDateRangeFormat)}',

                                                style: const TextStyle(

                                                    fontSize: 14),

                                              )

                                            ],

                                          ),

                                        ),

                                      ))

                                    ],

                                  ))));

                    }

                  });

        });

  }

}

**Lib/utils/constants.dart:**

const kAppBarDateFormat = 'M/yyyy';

const kMonthFormat = 'MMMM';

const kMonthFormatWidthYear = 'MMMM yyyy';

const kDateRangeFormat = 'dd-MM-yy';

**Lib/utils/extensions.dart:**

import 'package:intl/intl.dart';

extension DateTimeExt on DateTime {

  String format(String formatPattern) => DateFormat(formatPattern).format(this);

}

**Lib/res/colors.dart:**

import 'package:cr\_calendar\_example/widgets/event\_widget.dart';

import 'package:flutter/cupertino.dart';

const violet = Color(0xff7F51F5);

/// Цвета для [EventWidget].

const eventColors = [

  Color(0xff82D964),

  Color(0xffE665FD),

  Color(0xffF7980B),

  Color(0xfff2d232),

  Color(0xffFC6054),

  Color(0xffBEBEBE),

];

**Lib/pages/calendar\_page.dart:**import 'package:cr\_calendar/cr\_calendar.dart';

import 'package:cr\_calendar\_example/res/colors.dart';

import 'package:cr\_calendar\_example/utills/constants.dart';

import 'package:cr\_calendar\_example/utills/extensions.dart';

import 'package:cr\_calendar\_example/widgets/create\_event\_dialog.dart';

import 'package:cr\_calendar\_example/widgets/day\_events\_bottom\_sheet.dart';

import 'package:cr\_calendar\_example/widgets/day\_item\_widget.dart';

import 'package:cr\_calendar\_example/widgets/event\_widget.dart';

import 'package:cr\_calendar\_example/widgets/week\_days\_widget.dart';

import 'package:flutter/material.dart';

/// Класс CalendarPage представляет главную страницу приложения, на которой расположен календарь и управляющие элементы.

///

/// Календарь реализован с использованием библиотеки cr\_calendar. Он отображает дни месяца, разделяет их по неделям и позволяет пользователю просматривать события, связанные с конкретными днями.

///

/// Класс \_CalendarPageState содержит логику управления календарем и обработки событий пользователя. В методе initState происходит инициализация календаря и добавление примеров событий. В методе dispose происходит очистка ресурсов, занятых календарем.

///

/// Основные методы класса \_CalendarPageState включают:

///

/// \_changeCalendarPage - переход на следующий или предыдущий месяц календаря

/// \_onCalendarPageChanged - обработка смены месяца календаря

/// \_setTexts - установка текста заголовка на основе выбранного месяца и года

/// \_showCurrentMonth - переход к текущему месяцу

/// \_addEvent - открытие диалогового окна для создания нового события

/// \_createExampleEvents - добавление примеров событий в календарь

/// \_showDayEventsInModalSheet - отображение модального окна с событиями конкретного дня календаря.

/// Кроме того, класс \_CalendarPageState определяет виджеты, используемые для отображения элементов календаря, такие как DayItemWidget, WeekDaysWidget и EventWidget.

///

/// В целом, этот код позволяет создавать календарное приложение с использованием Flutter и библиотеки cr\_calendar.

class CalendarPage extends StatefulWidget {

  const CalendarPage({super.key});

  @override

  \_CalendarPageState createState() => \_CalendarPageState();

}

class \_CalendarPageState extends State<CalendarPage> {

  final \_currentDate = DateTime.now();

  late CrCalendarController \_calendarController;

  late String \_appbarTitle;

  late String \_monthName;

  @override

  void initState() {

    \_setTexts(\_currentDate.year, \_currentDate.month);

    \_createExampleEvents();

    super.initState();

  }

  @override

  void dispose() {

    \_calendarController.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      resizeToAvoidBottomInset: false,

      appBar: AppBar(

        centerTitle: false,

        title: Text(\_appbarTitle),

        actions: [

          IconButton(

            tooltip: 'Delete all events',

            icon: const Icon(Icons.delete),

              onPressed: () {

                setState(() {

                  \_calendarController.removeAllEvents();

                });

              }

          ),

          IconButton(

            tooltip: 'Go to current date',

            icon: const Icon(Icons.calendar\_today),

            onPressed: \_showCurrentMonth,

          ),

        ],

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: \_addEvent,

        child: const Icon(Icons.add),

      ),

      body: Column(

        children: [

          /// Строка управления календарем.

          Container(

            color: Color(0xFF212F3D),

            child:Row(

              mainAxisAlignment: MainAxisAlignment.spaceBetween,

              children: [

                IconButton(

                  icon: const Icon(Icons.arrow\_back\_ios),

                  onPressed: () {

                    \_changeCalendarPage(showNext: false);

                  },

                ),

                Text(

                  \_monthName,

                  style: const TextStyle(

                      fontSize: 16, color: violet, fontWeight: FontWeight.w600),

                ),

                IconButton(

                  icon: const Icon(Icons.arrow\_forward\_ios),

                  onPressed: () {

                    \_changeCalendarPage(showNext: true);

                  },

                ),

              ],

            ),

          ),

          /// Просмотр календаря.

          Expanded(

            child: CrCalendar(

              backgroundColor: Color(0xFF212F3D),

              firstDayOfWeek: WeekDay.monday,

              eventsTopPadding: 32,

              initialDate: \_currentDate,

              maxEventLines: 3,

              controller: \_calendarController,

              forceSixWeek: true,

              dayItemBuilder: (builderArgument) =>

                  DayItemWidget(properties: builderArgument),

              weekDaysBuilder: (day) => WeekDaysWidget(day: day),

              eventBuilder: (drawer) => EventWidget(drawer: drawer),

              onDayClicked: \_showDayEventsInModalSheet,

              minDate: DateTime.now().subtract(const Duration(days: 1000)),

              maxDate: DateTime.now().add(const Duration(days: 180)),

            ),

          ),

        ],

      ),

    );

  }

  /// Управление календарем с помощью кнопок со стрелками.

  void \_changeCalendarPage({required bool showNext}) => showNext

      ? \_calendarController.swipeToNextMonth()

      : \_calendarController.swipeToPreviousPage();

  void \_onCalendarPageChanged(int year, int month) {

    setState(() {

      \_setTexts(year, month);

    });

  }

  /// Установка текста панели приложения и название месяца поверх календаря.

  void \_setTexts(int year, int month) {

    final date = DateTime(year, month);

    \_appbarTitle = date.format(kAppBarDateFormat);

    \_monthName = date.format(kMonthFormat);

  }

  /// Показать страницу текущего месяца.

  void \_showCurrentMonth() {

    \_calendarController.goToDate(\_currentDate);

  }

  /// Показать [CreateEventDialog] с настройками для нового события.

  Future<void> \_addEvent() async {

    final event = await showDialog(

        context: context, builder: (context) => const CreateEventDialog());

    if (event != null) {

      \_calendarController.addEvent(event);

    }

  }

  void \_createExampleEvents() {

    final now = \_currentDate;

    \_calendarController = CrCalendarController(

      onSwipe: \_onCalendarPageChanged,

      events: [

      ],

    );

  }

  void \_showDayEventsInModalSheet(

      List<CalendarEventModel> events, DateTime day) {

    showModalBottomSheet(

        shape: const RoundedRectangleBorder(

            borderRadius: BorderRadius.vertical(top: Radius.circular(8))),

        isScrollControlled: true,

        context: context,

        builder: (context) => DayEventsBottomSheet(

              events: events,

              day: day,

              screenHeight: MediaQuery.of(context).size.height,

            ));

  }

}

**main.dart:**