Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Отчёт по:

ОТЧЕТ ПО ПРОИЗВОДСТВЕНО-ЭКСПЛУАТАЦИОННОЙ ПРАКТИКЕ

Выполнил студент 595 группы:
_____ Д. В. Осипенко
«___» _____ 2022 г.

Проверил: ст. преп. каф. ВТиЭ.
_____ И. А. Шмаков
«___» _____ 2022 г.

Барнаул 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Осуществлено прохождение производственно-эксплуатационной практике (далее - практика) направленния подготовки «Информатика и вычислительная техника» на базе кафедры Вычислительной техники и электроники (ВТиЭ) Алтайского государственного университета (АлтГУ). Периуд прохождения практики 4 недели: с 16 мая, по 11 июня 2022 года. за этот периуд мной произведено:

1. Ознакомление с нормативно-правовой базой, регламентирующей деятельность программиста на месте практики:
   1.1. Должностная инструкция программиста
   1.2. Инструкция по охране труда для программиста
   1.3. Вводный инструктаж
   1.4. Инструктаж по технике безопасности
2. Диагнотика компьтеров на работоспособность, выявление технических проблем.
3. Установка и настройка программного обеспечения для операционных систем Windows XP, 10 и Linux Ubuntu.
4. Разработка скриптов для bash и powershell
5. Разработка мобильного приложения на тему «Устные вычисления»
6. Руководство группой людей

Цель практики

1. Получить знания и навыки их практического применения в сфере системного администрирования и программирования
2. Ознакомится со спецификой деятельности системного администратора
3. Развитие лидерских качеств

Задачи практики

1. Поиск и изучение руководств по инсталляции, настройке, наладке, использованию программно-аппаратного обеспечения вычислительной техники, информационных и автоматизированных систем;

2. Освоение методик использования необходимого программного обеспечения;
3. Проверка работоспособности типовых узлов и устройств;
4. Использование программного обеспечения для решения практических задач, составление схем приема-передачи данных.

1. ГЛАВА. ОБЯЗОНОСТИ И ДЕЯТЕЛЬНОСТЬ ПРОИЗВОДИМЫЕ НА ПРЕДПРИЯТИИ

1.1. Диагностика компьютеров. Устранение найденныйх проблем.

Диагностика компьютера направленна на выявление проблем в его работе. В нашем случае план тестирования следующий:

1. Попытка запуска
2. Проверка подключенных устройств в BIOS
3. Проверка настроек BIOS
4. Проверка работоспособности RAM с помощью программы Memory Test 86 (memtest86) с загрузочной флешки
5. Проверка HDD на наличие битых секторов с помощью * с загрузочной флешки

В общем случае работа была проведена с 10тью компьютерами, которые в последующем были установленны в 206 аудиторию:

- Для двух ПК была произведена замена/установка блока питанию
- У 4 ПК была заменена батарейка BIOS
- Для 5 ПК устранена проблема с ошибкой объема памяти для дискетоприемника путем изменения значения в BIOS
- У 1 ПК устранена проблема с S.M.A.R.T. (self-monitoring, analysis and reporting technology) при загрузке компьютера путем отключения параметра в BIOS
- У 5 ПК установлено верное значение даты и времени
- Для каждого компьютера установленны RAM примерно на 1гб (в сумме)
- Успешно протестированно 9 ПК на наличие ошибок RAM, 10 на проблем с HDD
- У 1 ПК выявленны проблеммы с разъемом оперативной памяти материнской платы
- Обнаружены две неисправные плашки оперативной памяти

1.2. Установка Linux Ubuntu. Установка необходимого и удаление предустановленного/ненужного программного обеспечения

Необходимо установить Linux на один диск с Windows для запуска с помощью Dual Boot. Логично предположить, что производить установку для каждого ПК по отдельности долго и утомительно, поэтому был выбран один ПК, на нем установленна Linux, произведенна настройка, установка всего необходимого программного обеспечения и после этого дополнительно подключается hdd от другого ПК и производится клонирование данных с помощью программы CloneZila.

Установка Ubuntu linux производилась с помощью стандартного GUI установщика, добавлены дистрибутивы АлтГУ, осуществленно разбиение свободной части диска на три раздела: загрузка (boot), виртуальная память (swap), домашний раздел (root/home). После завершения установки был отредактирован и запущен bash скрипт для скачивания необходимых программных покетов.

Для Windows XP была произведенна активация с помощью ключа. Установленны недостающие драйвера и необходимы программы в 202, 206, 208, 210 аудиториях.

Для Windows 10 был написан скрипт на языке оболочки PowerShell для отключения некоторых служб, удаление предустановленных программ (Xbox, Cortana, ...), увелечения виртуальной памяти. Скрипт опробован и использован в 208 и 210 аудиториях.

1.3. Руководство приставленными студентами и первокусниками

Со второй недели прохождения практики ко мне были приставленны студенты Колледжа АлтГУ, проходящие практику на базе кафедры ВТиЭ. Мной была осуществлена помощь и обучение в проделывание перечисленных выше действиях, распределение обязанностей и раздача указаний при выполнении некоторых действий, например: установка компьютеров на рабочие места; контроль над первокурсниками.

С четвертой недели к подопечным присоеденились первокурсники, проходящие ознакомительную практику. В отношении их выполнялись надзор над выполнением поставленных задач руководством, помощь в разрешение трудностей, контроль посещаемости.

2. ГЛАВА. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ. ПРИЛОЖЕНИЕ ДЛЯ ТРЕНИРОВКИ УСТНЫХ ВЫЧИСЛЕНИЙ

В качестве индивидуального задания было необходимо написать мобильное приложение для тренировки устных вычисленний. Разработку мобильного приложения можно произвести с помощью двух способов:

1. Нативная разработка
2. Кроссплатформенная разработка

Нативная разработка

Нативная разработка делится на основе двух платформ: Android с языком Kotlin/Java и IOS на Swift/Object-c.

К достоинтсвам данного подхода можно отнести возможность написание нативного кода для каждой операционной системы, что позволяет иметь значительное преймущество в производительности и потенциальной оптимизированности кода. Так же к преймуществам можно отнести добротную документацию и поддержку со стараны вендоров: Google и Apple.

К недостаткам данного способа относятся скорость разработки и невозможность использовать одну кодовую базу для двух видов ОС, что при желании выйти на два рынка может возникнуть необходимость писать одно и тоже приложения для каждой платформы, что выливается в дополнительные затраты на разработку, тестирование.

Данный метод наиболее подходит для приложений, требующих производить множество вычисленний и возможному прямому доступу к возможностям платформы. Примеры таких приложений: Игры, нативные приложения (основанные на функционале платформы).

Кроссплатформенная разработка

Лидерами кроссплатформенного рынка являются React Nativ - расширение популярного web-based framework, разработаного компанией Facebook (запрещенной на территории Российской Федерации), призванного использовать написанный раннее код для web-приложения на мобилке. Главным преймуществом является я.п. JavaScript и ОГРОМНОЕ количество frontend-

разработчиков, разного уровня, освоивших данный framework. К недостаткам относится малая производительность скомпилированного приложения.

Другим вариантом, стремительно набирающем популярность является Flutter Framework, разработанный компанией Google, направленный для создание приложений на мобильных устройствах(IOS, Android), ПК(Linux, Windows) и веб страниц с использованием одной базы кода. Отличается относительно высокой производительностью, схожем способом постраения UI с React, быстрорастущим количеством сторонних библиотек и сообщества. Недостатками можно назвать саму компанию Google, которая своеобразно относится к своим проектам (может спокойно тянуть неудачные и не востребованные проекты, забить на потенциально интересные и успешние идеи и пустить все на ”свалку проектог гугл”) и сложность в отделение бизнес логики от кода ui.

В итоге мной выбор пал в сторону Flutter из-за высокой производительности (по сравнению с React nativ) и желанием в будущем выпустить приложение на Android и IOS.

2.1. Проектирование

Процесс разработки я начал с проектирования приложения. Первым делом я сделал наброски приложения, чтобы определится с функционалом и внешним видом

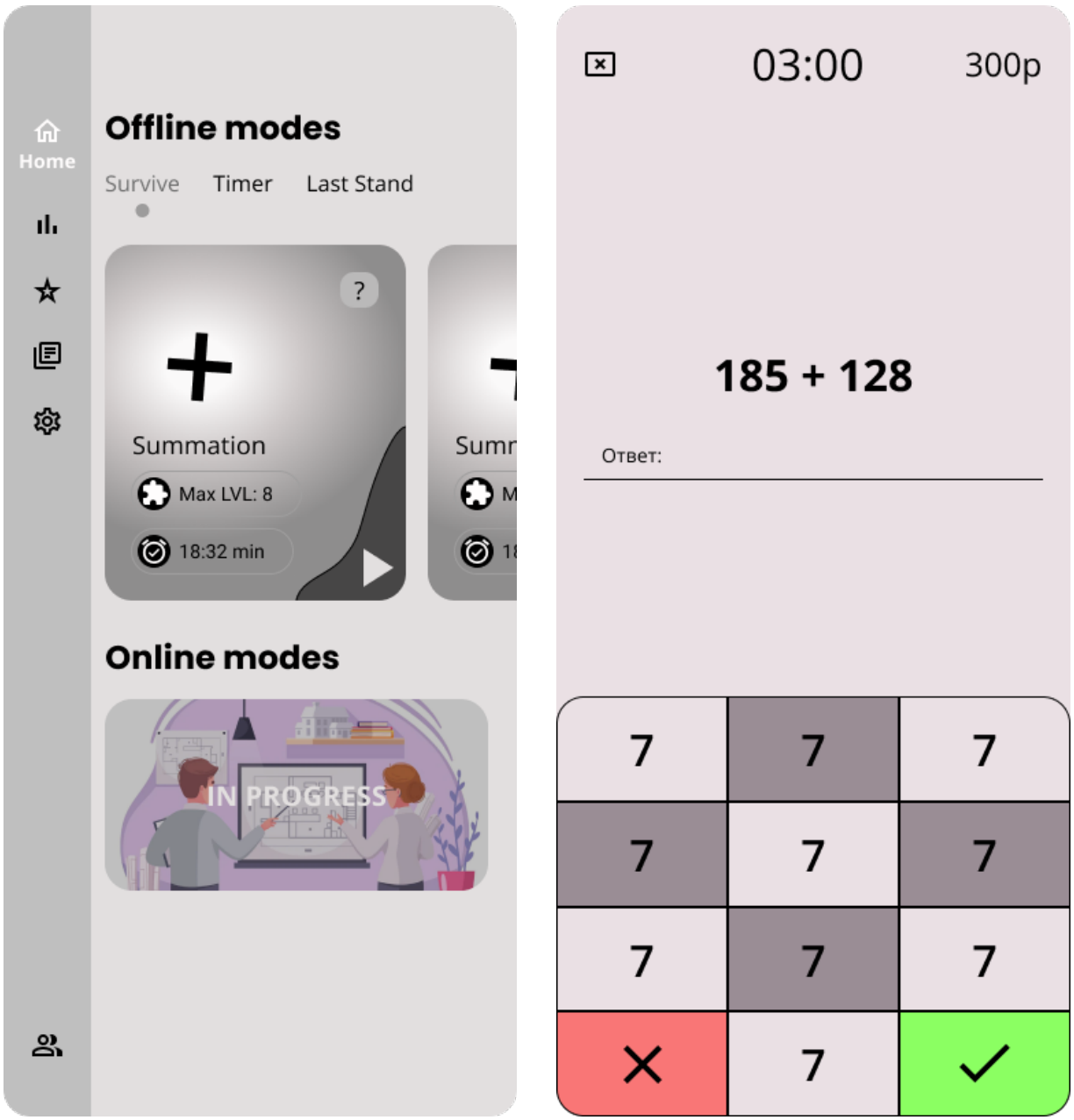

Рис 1. Экраны домашней страницы и процесса игры

Далее было решено использовать библиотеку контроля состояний Riverpod, для последующий реализации архитектуры MVC(P) (model view controller (provider)) для отделения бизнес логики от графического интерфейса, но из-за возникших трудностей с dependency injection (передача состояния и значений через дочерние (вложенные) элементы(виджеты)) и трудности создание новых объектов provider при переходе к игровму экрану, основанные на глобальном состоянии providers, окончательный выбор был сделан на библиотеку BLoC, использующие принцыпы bloc, cubit, state(состояние), event(событие)

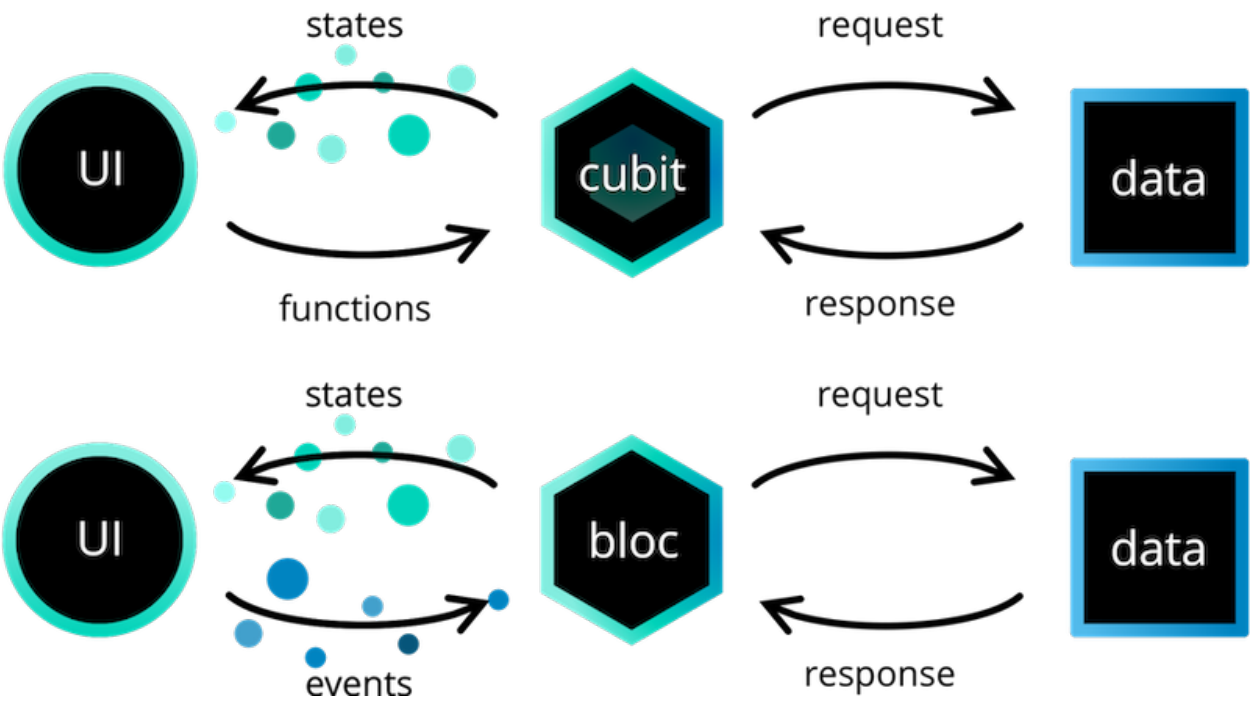

Рис. 2 Пример BLoC архитектуры с использованием bloc и cubit

2.2. Реализация

Проект представляет из себя 2 экрана: домашний и игровой. Так же имеется вложенный экран для домашнего экрана, необходимы для выбора режима игры. Для каждого экрана осуществленно извлечение бизнес логики путем использования Bloc элементов, состояний и событий.

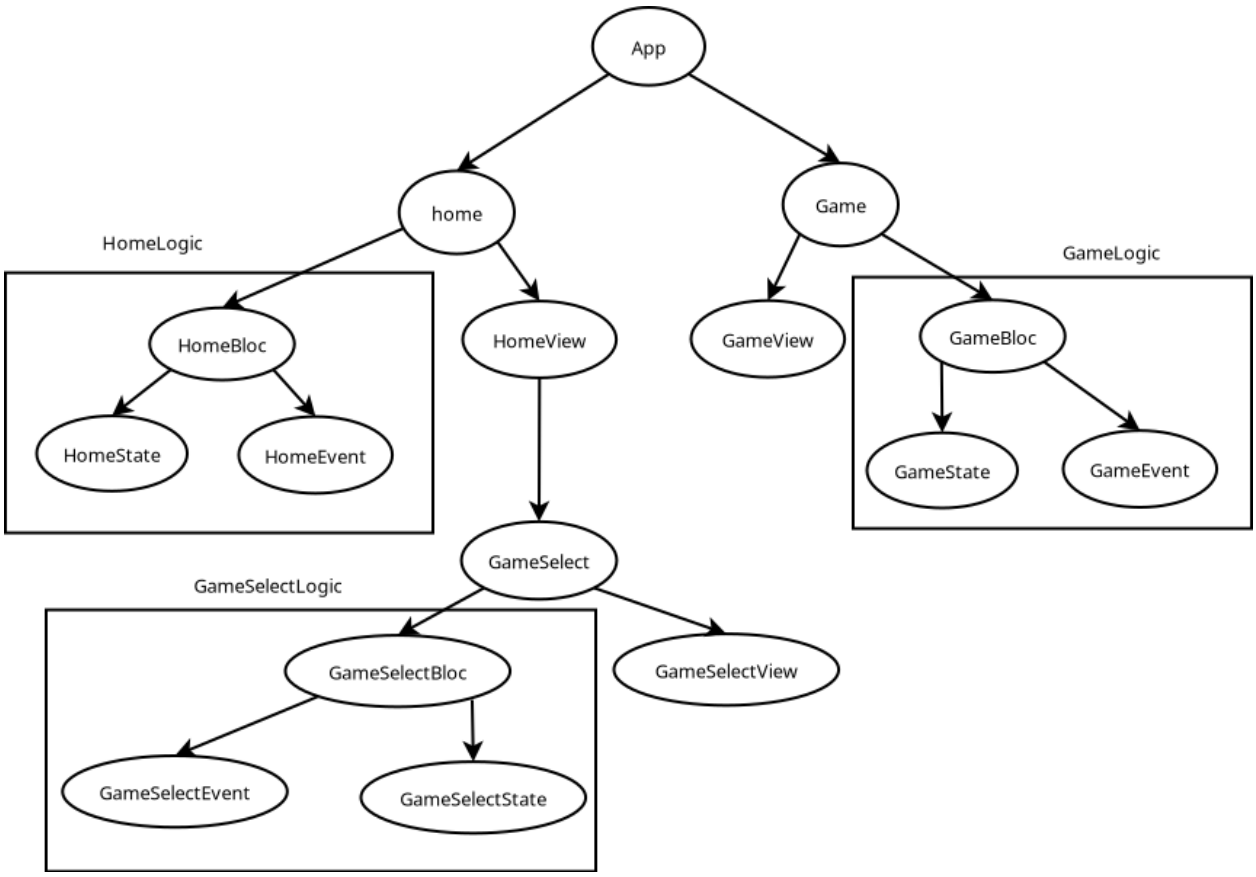

Рис. 3 Структура проекта.

ЗАКЛЮЧЕНИЕ

1. Пример ссылки на литературу [1].
2. Пример ссылки на литературу [2].
3. Пример ссылки на литературу [3].

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. [Электронный ресурс] Bitbucket — Википедия. — URL: https : / / ru . wikipedia.org/wiki/Bitbucket (дата обр. 28.03.2020).
2. [Электронный ресурс] Id Software — Википедия. — URL: https : / / ru . wikipedia.org/wiki/Id_Software (дата обр. 31.03.2020).
3. [Электронный ресурс] GitHub — Википедия. — URL: https://ru.wikipedia. org/wiki/GitHub (дата обр. 28.03.2020).

ПРИЛОЖЕНИЕ В

Листинг 2.1 Код приложения

```dart
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:smct/gameselect/gameselect.dart';
import 'package:smct/home/home.dart';

void main() {
  runApp(MultiBlocProvider(providers: [
    BlocProvider<HomeBloc>(create: (_) => HomeBloc()),
    BlocProvider<GameSelectBloc>(create: (_) =>
    ↪ GameSelectBloc()),
  ], child: const MentalCalc()));
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomeView(),
      theme: ThemeData.from(
        colorScheme: const ColorScheme(
          brightness: Brightness.dark,
          // Primary
          primary: Color(0xffFFB77E),
          onPrimary: Color(0xff4F2500),
          primaryContainer: Color(0xff703800),
          onPrimaryContainer: Color(0xffFFDCC2),
```

```
30            // Secondary
31            secondary: Color(0xffE3BFA6),
32            onSecondary: Color(0xff422B1A),
33            secondaryContainer: Color(0xff5A412F),
34            onSecondaryContainer: Color(0xffFFDCC2),
35            // Tertiary
36            tertiary: Color(0xffC6CA95),
37            onTertiary: Color(0xff2F330B),
38            tertiaryContainer: Color(0xff464A21),
39            onTertiaryContainer: Color(0xffE3E7AF),
40            // Error
41            error: Color(0xffFFB4A9),
42            onError: Color(0xff680003),
43            errorContainer: Color(0xff930006),
44            onErrorContainer: Color(0xffFFDAD4),
45            // Background
46            background: Color(0xffECE0DA),
47            onBackground: Color(0xff201A17),
48            surface: Color(0xffECE0DA),
49            onSurface: Color(0xff201A17),
50          ),
51          useMaterial3: true,
52          textTheme: TextTheme(
53              displaySmall: GoogleFonts.oswald(
54                fontSize: 36,
55                fontWeight: FontWeight.w400,
56              ),
57              headlineMedium: GoogleFonts.oswald(
58                fontSize: 28,
59                fontWeight: FontWeight.w400,
60              ),
61              labelLarge: GoogleFonts.openSans(
62                fontSize: 14,
63                fontWeight: FontWeight.w500,
```

```
                )),
        ),
      );
    }
}
```

```
import 'package:equatable/equatable.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:smct/gameselect/gameselect.dart';
import 'package:smct/widgets/widgets.dart';

part 'homestate.dart';
part 'homeevent.dart';
part 'homebloc.dart';
part 'homeview.dart';
```

```
part of 'home.dart';

class HomeView extends StatelessWidget {
  HomeView({Key? key}) : super(key: key);
  final hub_pages = [
    GameSelectView(),
    Center(
      child: Text('leaderboard'),
    ),
    Center(
      child: Text('tutorials'),
    ),
    Center(
      child: Text('Profile'),
    ),
  ];
```

```
17
18    @override
19    Widget build(BuildContext context) {
20      return Scaffold(
21        backgroundColor:
      ↪   Theme.of(context).colorScheme.background,
22        body: SafeArea(
23          child: Container(
24            height: MediaQuery.of(context).size.height,
25            width: MediaQuery.of(context).size.width,
26            child: Row(children: [
27              SideBar(),
28              SizedBox(
29                width: MediaQuery.of(context).size.width
      ↪   * .85,
30                height:
      ↪   MediaQuery.of(context).size.height,
31                child:
32                    BlocBuilder<HomeBloc,
      ↪   HomeState>(builder: (context, state) {
33                  return
      ↪   this.hub_pages[state.current_view];
34                }),
35              ),
36            ]),
37          ),
38        ),
39      );
40    }
41  }
```

```
1  part of 'home.dart';
2
3  class HomeBloc extends Bloc<HomeEvent, HomeState> {
```

```
4     HomeBloc() : super(const HomeInit()) {
5       on<HomeChangeView>(_changeView);
6     }
7
8     void _changeView(HomeChangeView event,
      ↪   Emitter<HomeState> emit) {
9       final view_id = event.view_id;
10      if (view_id >= 0) {
11        emit(HomeCurrentView(view_id));
12      }
13    }
14  }
```

```
1   part of 'home.dart';
2
3   abstract class HomeState extends Equatable {
4     final int current_view;
5
6     const HomeState(this.current_view);
7
8     @override
9     List<Object> get props => [this.current_view];
10  }
11
12  class HomeInit extends HomeState {
13    const HomeInit() : super(0);
14  }
15
16  class HomeCurrentView extends HomeState {
17    const HomeCurrentView(int view_id) : super(view_id);
18  }
```

```dart
part of 'home.dart';

abstract class HomeEvent extends Equatable {
  const HomeEvent();

  List<Object> get props => [];
}

class HomeChangeView extends HomeEvent {
  final int view_id;

  const HomeChangeView({required this.view_id});
}
```

```dart
import 'package:equatable/equatable.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:smct/game/game.dart';
import 'package:smct/widgets/widgets.dart';

part 'gameselectview.dart';
part 'gameselectstate.dart';
part 'gameselectbloc.dart';

enum OfflineType { Timer, Survive, Endurance }
```

```dart
part of 'gameselect.dart';

class GameSelectBloc extends Cubit<GameSelectState> {
  GameSelectBloc() : super(GameSelectInit()) {}

  void changeType(int index) {
    var type = OfflineType.values[index];
```

```
8       if (type.name != state.type.name) {
9         emit(GameSelectCurrent(type));
10      }
11    }
12  }
```

```
part of 'gameselect.dart';

abstract class GameSelectState extends Equatable {
  final OfflineType type;

  GameSelectState(this.type);

  @override
  List<Object> get props => [type];
}

class GameSelectInit extends GameSelectState {
  GameSelectInit() : super(OfflineType.Timer);
}

class GameSelectCurrent extends GameSelectState {
  GameSelectCurrent(OfflineType type) : super(type);
}
```

```
part of 'gameselect.dart';

class GameSelectView extends StatefulWidget {
  const GameSelectView({Key? key}) : super(key: key);

  @override
  State<StatefulWidget> createState() =>
  ↪  _GameSelectViewState();
```

```
}

class _GameSelectViewState extends State<GameSelectView>
    with SingleTickerProviderStateMixin {
  late TabController _tab_controller;
  final animations_list = [
    "assets/anim/plus.json",
    "assets/anim/minus.json",
    "assets/anim/plus.json",
    "assets/anim/div.json",
  ];
  @override
  void initState() {
    super.initState();
    this._tab_controller = TabController(length: 3, vsync: this);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        width: MediaQuery.of(context).size.width,
        height: MediaQuery.of(context).size.height,
        padding: EdgeInsets.only(top: 80, left: 20),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              "Offline modes",
              style:
    Theme.of(context).textTheme.displaySmall!.copyWith(
```

```
                    color:
↪ Theme.of(context).colorScheme.onBackground,
                  ),
            ),
            Container(
              height: 80,
              child: Align(
                alignment: Alignment.centerLeft,
                child: TabBar(
                  isScrollable: true,
                  controller: this._tab_controller
                    ..addListener(() {
                      context
                          .read<GameSelectBloc>()

↪ .changeType(this._tab_controller.index);
                    }),
                  indicator: DotIndicator(
                    color:
↪ Theme.of(context).colorScheme.primary,
                    distanceFromCenter: 20,
                    radius: 4,
                  ),
                  tabs: [
                    Tab(
                      text: "Timer",
                    ),
                    Tab(
                      text: "Survive",
                    ),
                    Tab(
                      text: "Endurance",
                    )
                  ],
```

```
                    ),
                  ),
                ),
                Container(
                  width: MediaQuery.of(context).size.width
    ↪  - 20,
                  height: 300,
                  child: ListView.builder(
                    scrollDirection: Axis.horizontal,
                    itemBuilder: (context, index) {
                      return ModeCard(
                        mode: Mode.values[index],
                        mode_anim:
    ↪  this.animations_list[index],
                        lvl: 8,
                        points: 30000,
                      );
                    },
                    itemCount: Mode.values.length,
                  ),
                )
              ],
            ),
          ),
      );
    }
}
```

```
import 'dart:async';
import 'dart:math';

import 'package:equatable/equatable.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
```

```dart
import 'package:smct/gameselect/gameselect.dart';

part 'gamemodel.dart';
part 'gameview.dart';
part 'gamestate.dart';
part 'gameevent.dart';
part 'gamebloc.dart';
part 'gamepage.dart';

enum Mode {
  Summation,
  Subtraction,
  Multiplication,
  Division,
}
```

```dart
part of 'game.dart';

class GameView extends StatelessWidget {
  GameView({Key? key}) : super(key: key);
  final TextEditingController txt_edit_controller =
      TextEditingController(text: '');

  final num_pad = ['7', '8', '9', '4', '5', '6', '1',
  ↪  '2', '3'];
  @override
  Widget build(BuildContext context) {
    var state = context.select((GameBloc bloc) =>
  ↪  bloc.state);
    var game = context.select((GameBloc bloc) =>
  ↪  bloc.state.game);
    var points = game.points;
    var left_value = game.left_value;
    var right_value = game.right_value;
```

```
16      var sign;
17      switch (game.mode) {
18        case Mode.Summation:
19          sign = '+';
20          break;
21        case Mode.Subtraction:
22          sign = '-';
23          break;
24        case Mode.Multiplication:
25          sign = '*';
26          break;
27        case Mode.Division:
28          sign = '/';
29          break;
30      }
31      var buttons_text_style = Theme.of(context)
32          .textTheme
33          .labelLarge!
34          .copyWith(color:
    ↪ Theme.of(context).colorScheme.onPrimary);
35      return Scaffold(
36        body: Column(
37          children: [
38            if (state is GameStateRunning) ...[
39              Container(
40                width: MediaQuery.of(context).size.width,
41                height: 80,
42                margin: EdgeInsets.only(top: 40, left:
    ↪ 20, right: 20),
43                child: Row(
44                  mainAxisAlignment:
    ↪ MainAxisAlignment.spaceBetween,
45                  crossAxisAlignment:
    ↪ CrossAxisAlignment.center,
```

```
                children: [
                  FloatingActionButton.small(
                    onPressed: () {
                      Navigator.pop(context);
                    },
                    child: Icon(Icons.clear),
                  ),
                  Container(
                    child: Text(
                      "$points p",
                      style: Theme.of(context)
                          .textTheme
                          .headlineMedium!
                          .copyWith(
                            color:
    ↪ Theme.of(context).colorScheme.onBackground,
                          ),
                    ),
                  ),
                ],
              ),
            ),
            Container(
              width: 120,
              height: 30,
              margin: EdgeInsets.only(bottom: 20),
              child: Row(
                mainAxisAlignment:
    ↪ MainAxisAlignment.center,
                crossAxisAlignment:
    ↪ CrossAxisAlignment.center,
                children: [
                  for (int i = 0; i < game.health; i++)
                    Icon(
```

```
                    Icons.heart_broken,
                    size: 28,
                    color:
    ↪ Theme.of(context).colorScheme.onPrimary,
                  )
                ],
              ),
            ),
            Text(
              "$left_value $sign $right_value",
              style:
    ↪ Theme.of(context).textTheme.displaySmall!.copyWith(
                  color:
    ↪ Theme.of(context).colorScheme.onBackground,
                ),
            ),
            Container(
              height: 120,
             width: MediaQuery.of(context).size.width,
              padding: EdgeInsets.symmetric(
                horizontal: 80,
              ),
              margin: EdgeInsets.only(top: 20),
              child: TextFormField(
                controller: this.txt_edit_controller,
                enabled: false,
                style:
    ↪ Theme.of(context).textTheme.headlineMedium!.copyWith(
                    color:
    ↪ Theme.of(context).colorScheme.onBackground,
                  ),
                decoration: InputDecoration(
                  prefixIcon: Row(
                    mainAxisSize: MainAxisSize.min,
```

```
                    mainAxisAlignment:
    ↪ MainAxisAlignment.center,
                    children: [
                      Text(
                        'Answer: ',
                        style: Theme.of(context)
                            .textTheme
                            .headlineMedium!
                            .copyWith(
                              color:
    ↪ Theme.of(context).colorScheme.onBackground,
                            ),
                      ),
                    ],
                  ),
                ),
              ),
            ),
            Expanded(child: SizedBox()),
            Container(
              width: MediaQuery.of(context).size.width,
              height:
    ↪ MediaQuery.of(context).size.height / 1.9,
              padding: const EdgeInsets.symmetric(
                horizontal: 50,
              ),
              child: GridView.count(
               physics: NeverScrollableScrollPhysics(),
                crossAxisCount: 3,
                crossAxisSpacing: 20,
                mainAxisSpacing: 10,
                children: [
                  for (var i in this.num_pad)
                    NumPadBtn(
```

```
                    txt_edit_controller:
    txt_edit_controller,
                    buttons_text_style:
    buttons_text_style,
                    text: i,
                  ),
                ElevatedButton(
                    onPressed: () {
                    this.txt_edit_controller.clear();
                    },
                    child: Icon(
                      Icons.clear,
                      color:
    Theme.of(context).colorScheme.onPrimary,
                    )),
                ElevatedButton(
                    onPressed: () {
                      this.txt_edit_controller.text
    += '0';
                    },
                    child: Text(
                      '0',
                      style: buttons_text_style,
                    )),
                ElevatedButton(
                  onPressed: () {
                    if (this.txt_edit_controller.text
    != '') {

    context.read<GameBloc>().add(GameEventSubmit(

    int.parse(this.txt_edit_controller.text)));
                    this.txt_edit_controller.clear();
                    }
```

```
164                     },
165                     child: Icon(
166                       Icons.clear,
167                       color:
    ↪ Theme.of(context).colorScheme.onPrimary,
168                     ),
169                   ),
170                 ],
171               ),
172             ),
173           ],
174           if (state is GameStateFinish) ...[
175             Container(
176               width: MediaQuery.of(context).size.width,
177               height:
    ↪ MediaQuery.of(context).size.height,
178               child: Column(
179                 mainAxisAlignment:
    ↪ MainAxisAlignment.center,
180                 crossAxisAlignment:
    ↪ CrossAxisAlignment.center,
181                 children: [
182                   Text(
183                     'Your points ${game.points}',
184                     style:
    ↪ Theme.of(context).textTheme.displaySmall!.copyWith(
185                           color:
    ↪ Theme.of(context).colorScheme.onBackground),
186                   ),
187                   SizedBox(
188                     height: 40,
189                   ),
190                   FloatingActionButton.extended(
191                     heroTag: null,
```

```
192                     onPressed: () {
193
    ↪ context.read<GameBloc>().add(GameEventStart());
194                     },
195                     elevation: 0,
196                     label: Text('Try again'),
197                     icon: Icon(Icons.play_arrow),
198                   ),
199                   SizedBox(
200                     height: 10,
201                   ),
202                   FloatingActionButton.extended(
203                     heroTag: null,
204                     onPressed: () {
205                       Navigator.pop(context);
206                     },
207                     label: Text('back'),
208                     icon: Icon(Icons.arrow_back),
209                     elevation: 0,
210                     backgroundColor:
    ↪ Theme.of(context).colorScheme.background,
211                     foregroundColor:
    ↪ Theme.of(context).colorScheme.onBackground,
212                   ),
213                 ],
214               ),
215             )
216           ],
217           if (state is GameStateInit) ...[
218             Container(
219               width: MediaQuery.of(context).size.width,
220               height:
    ↪ MediaQuery.of(context).size.height,
221               child: Column(
```

```
222                 mainAxisAlignment:
    ↪ MainAxisAlignment.center,
223                 crossAxisAlignment:
    ↪ CrossAxisAlignment.center,
224                 children: [
225                   FloatingActionButton.extended(
226                     heroTag: null,
227                     onPressed: () {
228
    ↪ context.read<GameBloc>().add(GameEventStart());
229                     },
230                     elevation: 0,
231                     label: Text('Start'),
232                     icon: Icon(Icons.play_arrow),
233                   ),
234                   SizedBox(
235                     height: 10,
236                   ),
237                   FloatingActionButton.extended(
238                     heroTag: null,
239                     onPressed: () {
240                       Navigator.pop(context);
241                     },
242                     label: Text('back'),
243                     icon: Icon(Icons.arrow_back),
244                     elevation: 0,
245                     backgroundColor:
    ↪ Theme.of(context).colorScheme.background,
246                     foregroundColor:
    ↪ Theme.of(context).colorScheme.onBackground,
247                   ),
248                 ],
249               ),
250             )
```

```
            ],
          ],
        ),
      );
    }
}

class NumPadBtn extends StatelessWidget {
  const NumPadBtn({
    Key? key,
    required this.txt_edit_controller,
    required this.buttons_text_style,
    required this.text,
  }) : super(key: key);

  final TextEditingController txt_edit_controller;
  final TextStyle buttons_text_style;
  final String text;

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
        onPressed: () {
          this.txt_edit_controller.text += text;
        },
        child: Text(
          text,
          style: buttons_text_style,
        ));
  }
}
```

```
part of 'game.dart';

```

```dart
class GameBloc extends Bloc<GameEvent, GameState> {
  late final Game _game;
  GameBloc(Game game) : super(GameStateInit(game)) {
    this._game = game;
    on<GameEventStart>(_onStart);
    on<GameEventSubmit>(_onSubmit);
  }

  void _onSubmit(GameEventSubmit event,
  ↪ Emitter<GameState> emit) {
    var game = state.game;
    if (event.answer == game.result) {
      game = game.incrementPoints();
      if (game.points % (5 * game.lvl) == 0) {
        game = game.changeLvl();
        game = game.changeMinMax();
      }
    } else {
      game = game.decrementPoints();
      game = game.decrementHealth();
    }
    if (game.health == 0) {
      emit(GameStateFinish(game));
    } else {
      game = game.generateValues();
      emit(GameStateRunning(game));
    }
  }

  void _onStart(GameEventStart event,
  ↪ Emitter<GameState> emit) {
    Game game = this._game.copyWith();
    game = game.generateValues();
    emit(GameStateRunning(game));
```

```
  }
}
```

```
part of 'game.dart';

abstract class GameState extends Equatable {
  final Game game;

  GameState(this.game);

  @override
  List<Object> get props => [game];
}

class GameStateInit extends GameState {
  GameStateInit(Game? game) : super(game ?? Game());
}

class GameStateRunning extends GameState {
  GameStateRunning(Game game) : super(game);
}

class GameStateFinish extends GameState {
  GameStateFinish(Game game) : super(game);
}
```

```
part of 'game.dart';

abstract class GameEvent extends Equatable {
  const GameEvent();

  @override
  List<Object> get props => [];
```

```
8  }
9
10 class GameEventStart extends GameEvent {}
11
12 class GameEventSubmit extends GameEvent {
13   final int answer;
14
15   GameEventSubmit(this.answer);
16 }
17
18 class GameEventEnd extends GameEvent {}
19
20 class GameEventRestart extends GameEvent {}
```