

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №2

по дисциплине «Разработка кроссплатформенных мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы ИКБО-32-22 | Козеев Василий Александрович |
| **Проверила:** | Зырянова С.А. |

МОСКВА 2025

**Практическая работа 2   
Установка Flutter и создания первого проекта**

**Тема**

Виджеты. Дерево виджетов. Stateless Widget.

**Задачи**

Задание 1

1) Выбрать тему (предметную область (ПО)) из предложенных ниже или свою и согласовать ее с преподавателем

2) Создать приложение для выбранной предметной области согласно примерному макету.

3) В отчет по практической работе внести программный код приложения, скриншот дерева виджетов приложения, скриншот результата.

**Практическая часть**

Тема – Игровые движки.

Изображение выглядит как текст, снимок экрана, меню, дизайн

Автоматически созданное описание

Скриншот 1 – Дерево виджетов приложения ч1

Изображение выглядит как текст, меню, снимок экрана, диаграмма

Автоматически созданное описание

Скриншот 2 – Дерево виджетов приложения ч2

Изображение выглядит как текст, программное обеспечение, веб-страница, Шрифт

Автоматически созданное описание

Скриншот 3 – Запуск в Windows

Изображение выглядит как снимок экрана, текст, мультимедиа, программное обеспечение

Автоматически созданное описание

Скриншот 4 – Запуск на Android

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Скриншот 5 – Запуск в web

Листинг 1 – main.dart:

|  |
| --- |
| import 'package:flutter/material.dart'; import 'models/game\_engine.dart'; import 'widgets/engine\_tab.dart';  void main() {  runApp(const GameEnginesApp()); }  class GameEnginesApp extends StatelessWidget {  const GameEnginesApp({super.key});   @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Игровые движки',  theme: ThemeData(  primarySwatch: Colors.*blue*,  appBarTheme: const AppBarTheme(  titleTextStyle: TextStyle(  fontSize: 20.0,  fontWeight: FontWeight.*bold*,  ),  ),  ),  home: const GameEnginesScreen(),  debugShowCheckedModeBanner: false,  );  } }  class GameEnginesScreen extends StatefulWidget {  const GameEnginesScreen({super.key});   @override  State<GameEnginesScreen> createState() => \_GameEnginesScreenState(); }  class \_GameEnginesScreenState extends State<GameEnginesScreen> with SingleTickerProviderStateMixin {  late TabController \_tabController;   @override  void initState() {  super.initState();  \_tabController = TabController(length: gameEngines.length, vsync: this);  }   @override  void dispose() {  \_tabController.dispose();  super.dispose();  }   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: const Text('ИГРОВЫЕ ДВИЖКИ'),  centerTitle: true,  bottom: TabBar(  controller: \_tabController,  isScrollable: true,  tabs: gameEngines.map((engine) => Tab(text: engine.name)).toList(),  ),  ),  body: TabBarView(  controller: \_tabController,  children: gameEngines.map((engine) => EngineTab(engine: engine)).toList(),  ),  );  } } |

Листинг 2 – engine\_tab.dart:

|  |
| --- |
| import 'package:flutter/material.dart'; import '../models/game\_engine.dart';  class EngineTab extends StatefulWidget {  final GameEngine engine;   const EngineTab({super.key, required this.engine});   @override  State<EngineTab> createState() => \_EngineTabState(); }  class \_EngineTabState extends State<EngineTab> {  int \_currentImageIndex = 0;   void \_cycleImage() {  setState(() {  \_currentImageIndex = (\_currentImageIndex + 1) % widget.engine.imageUrls.length;  });  }   @override  Widget build(BuildContext context) {  return SingleChildScrollView(  padding: const EdgeInsets.all(16.0),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text(  widget.engine.name,  style: const TextStyle(fontSize: 24, fontWeight: FontWeight.*bold*),  ),  const SizedBox(height: 10),  Text(  widget.engine.description,  style: const TextStyle(fontSize: 16),  ),  const SizedBox(height: 20),  // Новый макет: изображение слева, особенности справа  LayoutBuilder(  builder: (context, constraints) {  if (constraints.maxWidth > 600) {  // На широких экранах: изображение слева, список справа  return Row(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Expanded(  flex: 1,  child: \_buildImage(),  ),  const SizedBox(width: 20),  Expanded(  flex: 1,  child: \_buildFeatures(),  ),  ],  );  } else {  // На узких экранах: изображение сверху, список снизу  return Column(  children: [  \_buildImage(),  const SizedBox(height: 20),  \_buildFeatures(),  ],  );  }  },  ),  const SizedBox(height: 20),  // Блок с иконкой и информацией об авторе  \_buildAuthorInfo(),  ],  ),  );  }   Widget \_buildImage() {  return GestureDetector(  onTap: \_cycleImage,  child: Container(  padding: const EdgeInsets.all(8.0),  decoration: BoxDecoration(  border: Border.all(color: Colors.*grey*),  borderRadius: BorderRadius.circular(8.0),  ),  child: Image.network(  widget.engine.imageUrls[\_currentImageIndex],  height: 200,  fit: BoxFit.contain,  loadingBuilder: (context, child, loadingProgress) {  if (loadingProgress == null) return child;  return const Center(child: CircularProgressIndicator());  },  errorBuilder: (context, error, stackTrace) {  return const Center(  child: Icon(Icons.*error*, size: 50, color: Colors.*red*),  );  },  ),  ),  );  }   Widget \_buildFeatures() {  return Container(  padding: const EdgeInsets.all(12.0),  decoration: BoxDecoration(  color: Colors.*grey*[100],  borderRadius: BorderRadius.circular(8.0),  border: Border.all(color: Colors.*grey*[300]!),  ),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  const Text(  'Основные особенности:',  style: TextStyle(fontSize: 18, fontWeight: FontWeight.*bold*),  ),  const SizedBox(height: 10),  ...widget.engine.features.asMap().entries.map((entry) {  final index = entry.key;  final feature = entry.value;  return Padding(  padding: const EdgeInsets.only(bottom: 8.0),  child: Row(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Container(  width: 24,  height: 24,  alignment: Alignment.*center*,  decoration: BoxDecoration(  color: Colors.*blue*,  borderRadius: BorderRadius.circular(12),  ),  child: Text(  '${index + 1}',  style: const TextStyle(  color: Colors.*white*,  fontWeight: FontWeight.*bold*,  fontSize: 12,  ),  ),  ),  const SizedBox(width: 10),  Expanded(child: Text(feature)),  ],  ),  );  }),  ],  ),  );  }   Widget \_buildAuthorInfo() {  return Container(  padding: const EdgeInsets.all(12.0),  decoration: BoxDecoration(  color: Colors.*blue*[50],  borderRadius: BorderRadius.circular(8.0),  ),  child: Row(  children: [  // Иконка геймпада  Image.network(  'https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/Video-Game-Controller-Icon-IDV-green-industry.svg/800px-Video-Game-Controller-Icon-IDV-green-industry.svg.png',  width: 40,  height: 40,  errorBuilder: (context, error, stackTrace) {  return const Icon(Icons.*sports\_esports*, size: 40, color: Colors.*blue*);  },  ),  const SizedBox(width: 12),  const Expanded(  child: Text(  'Козеев В.А. ИКБО-32-22',  style: TextStyle(fontSize: 14, fontStyle: FontStyle.italic),  ),  ),  ],  ),  );  } } |

Листинг 3 – game\_engine.dart:

|  |
| --- |
| class GameEngine {  final String name;  final String description;  final List<String> imageUrls;  final List<String> features;   GameEngine({  required this.name,  required this.description,  required this.imageUrls,  required this.features,  }); }  // Данные о движках final List<GameEngine> gameEngines = [  GameEngine(  name: 'Unity',  description: 'Кроссплатформенный игровой движок для разработки 2D и 3D игр и приложений. Поддерживает множество платформ включая PC, мобильные устройства и игровые консоли.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/commons/thumb/1/19/Unity\_Technologies\_logo.svg/2560px-Unity\_Technologies\_logo.svg.png',  'https://media.sketchfab.com/models/2afd04b26ca64c6b8392f75e217ffdb9/thumbnails/b8b7fd406e624b27bcf5d22af1857ac0/6fc6917029464e18aef80d4f82f479a4.jpeg'  ],  features: [  'Визуальная среда разработки',  'Поддержка C# в качестве основного языка',  'Большое сообщество и активная экосистема',  'Бесплатный для индивидуальных разработчиков',  ],  ),  GameEngine(  name: 'Unreal Engine',  description: 'Мощный игровой движок с передовой графикой, используемый для создания AAA-игр и визуализаций. Разработан Epic Games.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/commons/thumb/c/c2/Unreal\_Engine\_Logo\_%28new\_typeface%29.svg/800px-Unreal\_Engine\_Logo\_%28new\_typeface%29.svg.png',  'https://bairesdev.mo.cloudinary.net/blog/2022/08/ue-logo-1400x788-1400x788-8f185e1e3635-1.jpg?tx=w\_1920,q\_auto'  ],  features: [  'Высококачественная графика с Ray Tracing',  'Система Blueprints для визуального программирования',  'Поддержка C++',  'Бесплатен с роялти после первого \$1 млн дохода',  ],  ),  GameEngine(  name: 'Godot',  description: 'Открытый и бесплатный игровой движок с поддержкой 2D и 3D графики. Имеет собственный язык программирования GDScript.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/commons/thumb/6/6a/Godot\_icon.svg/1200px-Godot\_icon.svg.png',  'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTF0JRsHg5TbN3vU\_PldPoFfyVWgwtKCG6KEw&s'  ],  features: [  'Полностью открытый исходный код',  'Собственный язык GDScript (похож на Python)',  'Поддержка множества платформ',  'Не требует роялти или подписки',  ],  ),  GameEngine(  name: 'CryEngine',  description: 'Игровой движок с продвинутой графикой, известный по играм серии Crysis. Разработан немецкой компанией Crytek.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/commons/8/8d/CryEngine\_Nex-Gen%284th\_Generation%29.png',  'https://upload.wikimedia.org/wikipedia/ru/9/93/CryEngine3\_logo.jpg'  ],  features: [  'Передовая технология глобального освещения',  'Мощные инструменты для создания ландшафтов',  'Поддержка виртуальной реальности',  'Платная модель на основе подписки',  ],  ),  GameEngine(  name: 'GameMaker Studio',  description: 'Игровой движок, ориентированный на 2D-разработку, с упором на простоту использования и быстрое создание прототипов.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/en/thumb/4/4b/GameMaker\_Logo.svg/1920px-GameMaker\_Logo.svg.png',  'https://upload.wikimedia.org/wikipedia/commons/2/2e/GameMaker-Studio-Logo1.jpg'  ],  features: [  'Визуальное программирование Drag & Drop',  'Собственный язык GML (GameMaker Language)',  'Быстрая разработка 2D игр',  'Экспорт на множество платформ',  ],  ),  GameEngine(  name: 'Construct',  description: 'Игровой движок, основанный на событиях, не требующий знания программирования. Идеален для начинающих разработчиков.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/commons/thumb/7/79/Construct\_3\_Logo.svg/800px-Construct\_3\_Logo.svg.png',  'https://upload.wikimedia.org/wikipedia/en/thumb/5/56/Construct\_2\_logo.png/250px-Construct\_2\_logo.png'  ],  features: [  'Программирование на основе событий',  'Не требует знаний программирования',  'Быстрое создание 2D игр',  'Экспорт в HTML5 и другие платформы',  ],  ),  GameEngine(  name: 'Source',  description: 'Игровой движок, разработанный Valve Corporation, известный по играм Half-Life 2, Counter-Strike: Source и другим.',  imageUrls: [  'https://upload.wikimedia.org/wikipedia/commons/thumb/6/67/Source\_engine\_logo\_and\_wordmark.svg/1920px-Source\_engine\_logo\_and\_wordmark.svg.png',  'https://www.sourcemodding.com/img/history/source2/source\_engine2\_logo.webp'  ],  features: [  'Продвинутая физика на основе Havok',  'Технология facial animation для реалистичной мимики',  'Интеграция с Steamworks',  'Модифицируемость и поддержка модов',  ],  ), ]; |