

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ  
И ИНФОРМАТИКИ

Кафедра технологий программирования

ЛАБОРАТОРНАЯ РАБОТА 10  
ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ МОБИЛЬНЫХ И  
ВСТРАИВАЕМЫХ СИСТЕМ»

Разработка мобильного приложения для ОС Автор с использованием  
Flutter SDK

Методические указания по выполнению лабораторной работы

Минск 2024

## СОДЕРЖАНИЕ

|  |    |
|--|----|
| Введение .....   | 3  |
| 1 Методические рекомендации.....                       | 4  |
| 1.1 Основной учебный материал .....                    | 4  |
| 1.2 Установка Flutter SDK для ОС Аврора .....          | 4  |
| 1.2.1 Установка Flutter SDK в Linux.....               | 5  |
| 1.3 Тестирование Flutter приложений для ОС Аврора..... | 6  |
| 2 Методические указания и задания.....                 | 7  |
| 2.1 Методические указания.....                         | 7  |
| 2.1.1 Критерии оценивания .....                        | 7  |
| 2.1.2 Отчет по лабораторной работе .....               | 7  |
| 2.1.2.1 Требования к репозиторию.....                  | 8  |
| 2.1.2.2 Защита отчета по лабораторной работе .....     | 9  |
| 2.2 Задания.....                                       | 10 |
| 2.2.1 Задание 1.....                                   | 10 |
| 2.2.2 Задание 2.....                                   | 11 |
| 2.2.2.1 Вариант 1.....                                 | 11 |
| 2.2.2.2 Вариант 2.....                                 | 12 |
| 2.2.2.3 Вариант 3.....                                 | 12 |
| 2.2.2.4 Вариант 4.....                                 | 12 |
| 2.2.2.5 Вариант 5.....                                 | 12 |
| 2.2.3 Контрольные вопросы.....                         | 13 |

## ВВЕДЕНИЕ

Целью работы является получению навыков разработки мобильных приложений на языке Dart в среде Aurora IDE или Aurora Studio с использованием Flutter SDK для мобильной операционной системы Аврора. Для достижения поставленной цели необходимо решить следующие задачи:

- установить пакет Aurora SDK, включающий Aurora IDE, Aurora SDK, виртуальные машины со средой сборки и эмулятором под управлением ОС Аврора или Aurora Studio и виртуальные машины со средой сборки и эмулятором под управлением ОС Аврора;
- установить Platform SDK;
- установить Flutter SDK;
- изучить примеры и разработать мобильное приложение согласно заданию.

## 1 Методические рекомендации

В данном разделе представлены рекомендации по выполнению лабораторной работы.

### 1.1 Основной учебный материал

Учебный материал для выполнения лабораторной работы изложен в источниках:

а) Платформа ОС Аврора:

- 1) 📖 Архитектура ОС Аврора;
- 2) 📖 Разработка прикладного программного обеспечения
- 3) 📖 Гайд по Aurora OS: как начать разработку для отечественной мобильной операционки
- 4) 📖 Как начать разрабатывать под ОС Aurora

б) Учебные материалы по разработке приложений для ОС Аврора с использованием Flutter SDK:

- 📖 Установка Flutter SDK для ОС Аврора в окружении Ubuntu
- 📖 Приложение на Flutter для ОС Аврора
- 📖 Flutter SDK для ОС Аврора
- 📖 Пошаговая инструкция по развертыванию и использованию Flutter SDK)

в) 📖 Simone Alessandria. Flutter Projects. — 2020. — 483 p.;


г) 📖 Richard Rose. Flutter and Dart Cookbook. — 2023. — 310 p..



### 1.2 Установка Flutter SDK для ОС Аврора


Мобильное приложение с использованием Flutter SDK с поддержкой ОС Аврора может быть разработано в любой среде разработки, позволяющей установить и подключить Flutter SDK.

К рекомендованным средам разработки относятся:

- Aurora IDE (входит в состав Aurora SDK).
- Aurora Studio (📖 <https://aurora.rt.ru>) — среда разработки на основе VS Code.




Если использовать среду разработки Aurora Studio или иную кроме Aurora IDE, то сборку проекта необходимо выполнить в командной строке. Для автоматизации процедур ручной сборки в консоли рекомендуется использовать пакет  Aurora CLI

На данный момент установка доступна на Linux и Windows через WSL2. Больше информации можно найти в разделе документации  «Установка» проекта  Flutter для ОС Аврора.

Работа на macOS доступна через удаленный доступ с серверу с Linux x86\_64, такую работу можно организовать как в примере проекта  «Настройка удаленного сервера Aurora OS».

### 1.2.1 Установка Flutter SDK в Linux

Операционная система Linux — основная среда для работы с Flutter для ОС Аврора в данный момент. Рекомендуемый дистрибутив — Ubuntu (22.04), работа на других возможна, но возможными проблемами конфигурации и сборки. Для настройки рабочего места и установки всех необходимых инструментов следует выполнить следующие шаги:

- а) Установить  Aurora SDK.
- б) Установить  Platform SDK.
- в) Установить  Flutter SDK.

*Aurora SDK* — это набор инструментов для создания Qt/QML приложений. Подробно установка Aurora SDK рассмотрена в лабораторной работе 9.

Aurora SDK содержит среди других компонентов эмулятор, на котором можно запускать приложения Flutter.

*Platform SDK* — предназначен для сборки приложений из командной строки и может использоваться в системах CI/CD. Этот компонент отвечает за сборку приложений в установочный RPM пакет. Сборка приложений осуществляется Platform SDK через Flutter CLI.




Flutter SDK — фреймворк с открытым исходным кодом который позволяет собрать приложение для разных платформ, в том числе и ОС Аврора. Фреймворк с поддержкой платформы ОС Аврора доступен в репозитории Flutter. Реализую поддержку платформы ОС Аврора фреймворк, для всех

платформ, остается полнофункциональным, то есть его можно использовать и для разработки других платформ.

Подробнее об установке Flutter SDK и других компонентов в документации  «Установка на Linux».

### 1.3 Тестирование Flutter приложений для ОС Аврора

Тестирование Flutter приложения может быть реализовано за счет использования QT Tests (см. лабораторную работу 9), инструментов тестирования программного средства Flutter или других решений.

Программное средство разработки Flutter предоставляет разработчику полный набор инструментов для тестирования приложений:  unit test (модульный тест),  widget test (тест виджетов) и  integration test (интеграционный тест).

Каждый тест решает задачу на своем уровне.

Модульный тест тестирует одну функцию, метод или класс. Его цель проверить правильность работы определенной функции, метода или класса. Внешние зависимости для тестируемого модуля обычно передаются как параметр.

Виджет тест тестирует один виджет. Цель такого теста — убедиться, что пользовательский интерфейс виджета выглядит и взаимодействует, как запланировано. Тестирование виджета происходит в тестовой среде, которая обеспечивает контекст жизненного цикла виджета. Также тестируемый виджет должен иметь возможность получать действия и события пользователя и отвечать на них.

Интеграционный тест тестирует все приложение или его большую часть. Цель интеграционного теста — убедиться, что все тестируемые виджеты и сервисы работают вместе, как ожидалось. Кроме того, вы можете использовать интеграционные тесты для проверки производительности вашего приложения. Как правило, интеграционный тест выполняется на реальном устройстве или эмуляторе.

Подробнее в документации  Testing Flutter apps.

## **2 Методические указания и задания**

### **2.1 Методические указания**

Общие рекомендации по выполнению заданий лабораторной работы:

- а) Проект для любого из заданий может быть реализован в виде мобильного приложения в среде Aurora IDE или Aurora Studio (или иной) с использованием Aurora SDK и Flutter SDK.
- б) Проект для заданий 2 и 3 должен содержать модульные тесты.
- в) Проект для каждого из заданий должен предусматривать обработку исключительных ситуаций.

#### **2.1.1 Критерии оценивания**

##### **Оценка 4-5**

Выполнено задание 1. Представлен отчет с ответами на контрольные вопросы, исходный код проекта в `git`-репозитории. Лабораторная работа сдана с задержкой в 1-2 недели.

##### **Оценка 6-7**

Выполнены задания 1-2. Функционал приложения в задании 2 соответствует требованиям. Разработаны для приложения в задании 2 модульные тесты (не менее 3-х) и один тест для тестирования виджета. Представлен отчет с ответами на контрольные вопросы, исходный код проектов в `git`-репозитории. Лабораторная работа сдана с задержкой в 1 неделю.

##### **Оценка 8-9**


Выполнены задания 1-2 на отличном уровне, задание 2 содержит дополнительный функционал. Разработаны модульные тесты (максимальное покрытие тестами) и тесты виджетов (не менее 3-х).

Представлен отчет, ответы на контрольные вопросы, исходные коды скриптов в `git`-репозитории. Отчет, исходный код не содержат ошибок. Лабораторная работа сдана в срок.

#### **2.1.2 Отчет по лабораторной работе**

Отчет по лабораторной работе должен быть опубликован в репозитории и отвечать требованиям:

1. Отчет по лабораторной работе состоит из письменного отчета, кода приложений и тестов к ним, опубликованных в репозиторий
2. Письменный отчет содержит цель работы.
3. Письменный отчет включает вариант задания.
4. Письменный отчет добавить описание ключевых моментов реализации и тестов.
5. Исходный код программ для каждого задания опубликовать в подкаталоге `/src` соответствующего каталога проекта (задания) и в соответствующей ветке репозитория.





Ссылка на репозиторий для лабораторной работы 10 доступна в курсе  «Программирование мобильных и встраиваемых систем».

В файле `README` в корневом каталоге проекта на *github* должна быть ссылка на отчёт и краткие сведения о выполненных заданиях (проектах).

Отчет опубликовать во внешнем хранилище или в репозитории в каталоге `/docs`. **Отчёт должен результаты тестов по каждому приложению и ответы на контрольные вопросы.**

### 2.1.2.1 Требования к репозиторию

Корневой каталог репозитория должен включать:

1. файл `README`, содержащий ссылку на отчет;
2. при публикации отчета в репозитории, разместить его в папке `/docs`;
3. каждое задание находится в каталоге проекта, структура которого соответствует требованиям задания;
4. в корневом каталоге репозитория и папках проектов должен быть добавлен файл `.gitignore`, в котором определены правила для исключения временных, исполняемых, объектных файлов и т.д. В качестве примера для проекта мобильного приложения рекомендуется использовать  шаблон `.gitignore` для среды QT и  шаблон `gitignore` для проектов Dart из библиотеки шаблонов  A collection of `.gitignore` templates на  *Github*. При необходимости шаблонный файл может быть дополнен инструкциями.

Пример оформления файла `README` может быть таким:

|   |  |
|---|--|
| 1 | <code># Overview</code>                    |
| 2 |  |
| 3 | <code>Отчет по лабораторной работе.</code> |



```

4
5 # Usage
6
7 // Заменить <<link>> и <<folder>> на соответствующие ссылки и названия
8
9 To check, please, preview report by <<link>> and script files in
  <<folder>>.
10
11 # Author
12
13 Your name and group number.
14
15 # Additional Notes
16
17 // СКОПИРОВАТЬ И ВСТАВИТЬ ССЫЛКУ
18 // НА СВОЙ РЕПОЗИТОРИЙ, НАПРИМЕР
19
20 https://github.com/maryiad/lab10-gr16-david

```

### 2.1.2.2 Защита отчета по лабораторной работе

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приёме работы у студента.

Выполнение студентом лабораторной работы и сдача её результатов преподавателю происходит следующим образом:

1. Студент выполняет разработку программ.

В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению для программ на языке Dart.

2. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности её работы, а также свои знания по теме лабораторной работы.

Исходные тексты программ должны соответствовать требованиям к оформлению, приведённым в приложении. Недопустимо отсутствие в тексте программы следующих важных элементов оформления: спецификации программного файла и подпрограмм, а также отступов, показывающих структурную вложенность языковых конструкций.

Для проверки правильности работы программы студенту необходимо разработать набор тестов и на их основе провести тестирование программы. Тестовый набор должен включать примеры входных и выходных данных,

приведённые в тексте задачи, а также тесты, разработанные студентом самостоятельно.

Самостоятельная проверка знаний по теме лабораторной работы выполняется с помощью контрольных вопросов и заданий, приведённых в конце текста лабораторной работы.

3. Студент защищает разработанные программы. Защита заключается в том, что студент должен ответить на вопросы преподавателя, касающиеся разработанной программы, и контрольные вопросы.

К защите необходимо представить исходные тексты программ, оформленных в соответствии с требованиями, исходные коды модульных тестов (unit-тестов).


При защите лабораторной работы студент демонстрирует исходный код теста (описание входных данных и соответствующих им выходных данных), описание выходных данных, полученных при запуске программы на данном тесте, и отметку о прохождении теста. Тест считается пройденным, если действительные результаты работы программы совпали с ожидаемыми.

Программы, не прошедшие тестирование, к защите не принимаются. В случае неверной работы программы хотя бы на одном тесте студент обязан выполнить отладку программы для поиска и устранения ошибки.


## **2.2 Задания**


### **2.2.1 Задание 1**

*Цель* — установить Flutter SDK и другие инструменты, познакомиться с примерами проектов и запустить приложения из примеров проекта в эмуляторе Аврора ОС.




1. Изучить рекомендации по установке Flutter SDK (см. 1.2. «Установка Flutter SDK для ОС Аврора») и на сайте  «Установка на Linux».


2. В лабораторной работе 9 установили программное средство разработки Aurora SDK. Установить Platform SDK и Flutter SDK, а так же другие инструменты, рекомендуемые в (см. 1.2. «Установка Flutter SDK для ОС Аврора»).

3. Изучить пример  «Приложение «Hello, World» на Flutter для ОС Аврора», скопировать код проекта, собрать и запустить в виртуальной машине ОС Аврора.

4. Изучить пример приложения  «Fluttery ToDo», включая тесты. Скопировать код проекта, собрать и запустить в виртуальной машине ОС Аврора.

### 2.2.2 Задание 2

Реализовать мобильное приложение на основе Flutter SDK для ОС Аврора согласно варианту, обеспечивающее локализацию, использование анимации (см. пример приложения  Animations), вывод прогноза погоды для выбранного объекта, построение маршрута на карте (см. пример  Place Tracker) и хранение данных в базе данных SQLite (см.  Persist data with SQLite):

- Структура приложения должна соответствовать одному из архитектурных шаблонов **Provider/Scoped Model** или **BLoC** (подробнее см.  Выбираем архитектуру для Flutter-приложения).
- Приложение должно быть локализовано на три языка, например русский, белорусский и английский.
- Содержит анимацию элементов пользовательского интерфейса (на выбор).
- Включает обработку исключительных ситуаций с выводом сообщений об ошибках (в консоль).
- При сборке проекта выполняются модульные тесты и тесты виджетов.

#### 2.2.2.1 Вариант 1

Разработать программу бронирования билетов на маршрутный автобус Гродненской области. Пользователь выбирает маршрут на карте, и ему предлагается список городов, при выборе города — список рейсов, включающих номер маршрутного автобуса, стоимость проезда и наличие свободных мест. При выборе города на карте пользователь может посмотреть прогноз погоды, получая данные от сервиса <https://openweathermap.org/api>. Данные должны храниться в базе данных sqlite.

#### **2.2.2.2 Вариант 2**

Разработать программу, интерактивную карту музеев города Витебска. Пользователь выбирает район города на карте, и ему предлагается список музеев в данном районе и выставки, которые в каждом музее проходят. При выборе района города на карте пользователь может посмотреть прогноз погоды, получая данные от сервиса <https://openweathermap.org/api>. Данные должны храниться в базе данных sqlite.

#### **2.2.2.3 Вариант 3**

Разработать программу бронирования билетов на самолет. Пользователь выбирает маршрут на карте, и ему предлагается список аэропортов Беларуси, при выборе аэропорта — список рейсов, включающих название рейса, стоимость проезда и наличие свободных мест. При выборе аэропорта на карте пользователь может посмотреть прогноз погоды, получая данные от сервиса <https://openweathermap.org/api>. Данные должны храниться в базе данных sqlite.

#### **2.2.2.4 Вариант 4**

Разработать программу, интерактивную карту вузов и колледжей города Бреста. Пользователь выбирает район города на карте, и ему предлагается список учебных заведений в данном районе. При выборе района города на карте пользователь может посмотреть прогноз погоды, получая данные от сервиса <https://openweathermap.org/api>. Данные должны храниться в базе данных sqlite.

#### **2.2.2.5 Вариант 5**

Разработать программу, интерактивную карту библиотек города Минска. Пользователь выбирает район города на карте, и ему предлагается список библиотек в данном районе. При выборе района города на карте пользователь может посмотреть прогноз погоды, получая данные от сервиса <https://openweathermap.org/api>. Данные должны храниться в базе данных sqlite.

### 2.2.3 Контрольные вопросы

1. Что такое Flutter?
2. Что такое Dart и почему он используется во Flutter?
3. Какие различные типы виджетов существуют во Flutter?
4. В чем разница между StatelessWidget и StatefulWidget во Flutter?
5. Для чего нужен ключ (Key) во Flutter?
6. Какие типы ключей используются во Flutter?
7. В чем разница между MaterialApp и WidgetsApp во Flutter?
8. Перечислите методы оптимизации приложений Flutter.
9. В чем разница между Navigator и Router во Flutter?
10. Что такое State в Flutter?
11. Для чего используется параметр BuildContext в методе build() виджета во Flutter?
12. В чем разница между методами push и pushReplacement во Flutter?
13. Для чего нужен инспектор виджетов во Flutter?
14. Для чего нужен виджет MediaQuery во Flutter?
15. Для чего нужен виджет SafeArea во Flutter?