САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет

по курсовому проекту

Дисциплина

«Проектирование мобильных приложений»

Выполнили:

Сергеев И.К.

группа: 33531/2

преподаватель:

Алексюк Артем Олегович

Санкт-Петербург

2018

Оглавление

[Список Иллюстраций 2](#_Toc533505572)

[Введение 3](#_Toc533505573)

[Платформа Flutter 4](#_Toc533505574)

[Dart 5](#_Toc533505575)

[Приложение 6](#_Toc533505576)  
[Главная страница 6](#_Toc533505577)  
[Страница группы 7](#_Toc533505578)  
[Обращение к Java - методу 8](#_Toc533505579)

[Интерфейс приложения 9](#_Toc533505580)

[Тестирование и отладка 10](#_Toc533505581)  
[Flutter analyze 10](#_Toc533505582)  
[Тесты 10](#_Toc533505583)

[Вывод 12](#_Toc533505584)

[Список литературы 12](#_Toc533505585)

# Список Иллюстраций

Рис.1. Структура Flutter 4

Рис.2. Краткая схема наследования виджетов 5

Рис.3. Календарь на неделю 8

Рис.4. Главная страница 9

Рис.5. Страница группы 9   
Рис.6. Горизонтальная ориентация 9   
Рис.6. Возможности Flutter Inspector 11

# Введение

Темой данного курсового проекта является написание мобильной версии для Web-приложения polytable.ru. Нами (мной и студентом группы 33531/2, Графовым Денисом) были обозначены два требования для реализуемого приложения: красивый и плавный графический интерфейс, а также единая кодовая база для Android и IOS. Именно для выполнения этих требований в качестве языка программирования, а также используемой библиотеки были выбраны Dart и Flutter. Далее в отчёте будут приведены некоторые особенности данных инструментов разработки.

# Платформа Flutter

**Flutter** — молодая, но очень многообещающая платформа, уже привлекшая к себе внимание крупных компаний, которые запустили свои приложения. Интересна эта платформа своей простотой сравнимой с разработкой веб-приложений, и скоростью работы наравне с нативными приложениями. Высокая производительность приложения и скорость разработки достигается за счет нескольких техник:

* В отличии от многих известных на сегодняшний день мобильных платформ, Flutter не использует JavaScript ни в каком виде. В качестве языка программирования для Flutter выбрали Dart, который компилируется в бинарный код, за счет чего достигается скорость выполнения операций сравнимая с Objective-C, Swift, Java, или Kotlin.
* Flutter не использует нативные компоненты, опять же, ни в каком виде, так что не приходится писать никаких прослоек для коммуникации с ними. Вместо этого, подобно игровым движкам, он отрисовывает весь интерфейс самостоятельно. Кнопки, текст, медиа-элементы, фон — все это отрисовывается внутри графического движка в самом Flutter.
* Для построения UI во Flutter используется декларативный подход, вдохновленный веб-фреймворком ReactJS, на основе виджетов (в мире веба именуемых компонентами). Для еще большего прироста в скорости работы интерфейса виджеты перерисовываются по необходимости — только когда в них что-то изменилось (подобно тому как это делает Virtual DOM в мире веб-фронтенда).

Давайте рассмотрим структуру данной библиотеки.

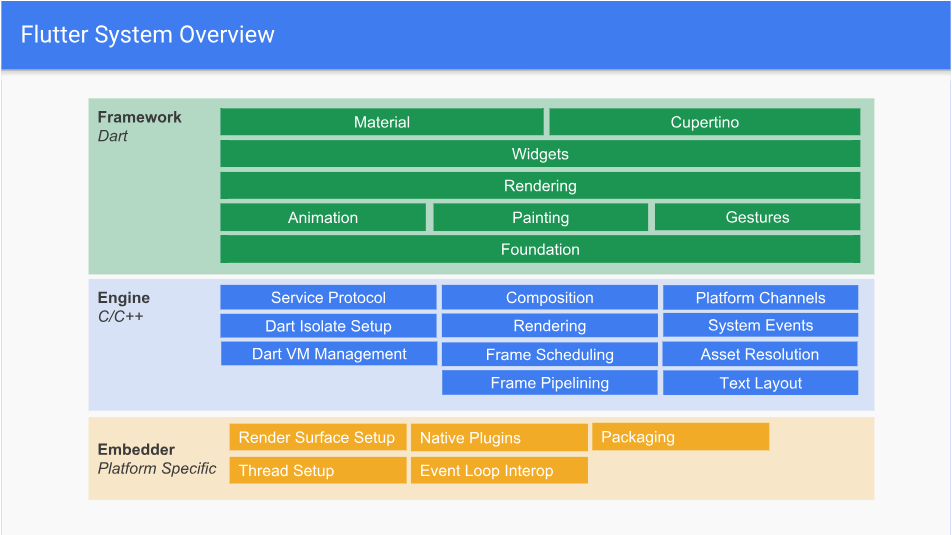


Рис.1. Структура Flutter

Верхний слой – наиболее используемый, это непосредственно элементы UI: виджеты, жесты, анимации и т.д. Средний слой – движок Skia, написанный на языке C++ и используемый для рендера описанных элементов. Нижний слой – взаимодействие с платформой, например, каналы для вызова методов, написанных на «родном» для платформы языке.

Чуть более подробно остановимся на верхнем слое, а именно на виджетах. Виджеты являются основными строительными блоками пользовательского интерфейса приложения Flutter. Каждый виджет является неизменным объявлением части пользовательского интерфейса. В отличие от других платформ, которые разделяют представления, контроллеры представлений, макеты и другие свойства, Flutter имеет согласованную унифицированную объектную модель: виджет. Они разделяются на Stateful и Stateless (хранящие и не хранящие состояние).

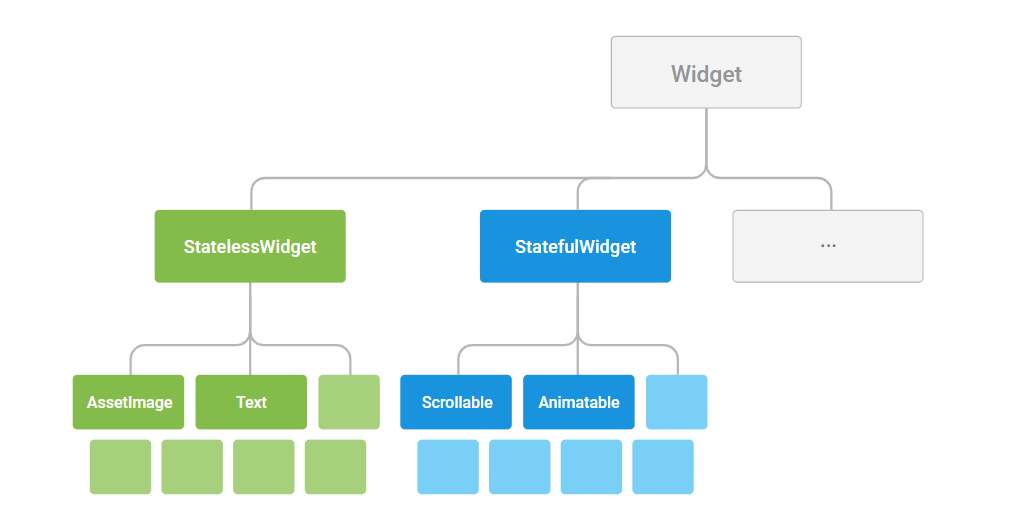


Рис.2. Краткая схема наследования виджетов

# Dart

**Dart** — язык программирования, созданный Google. Dart позиционируется в качестве замены/альтернативы [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). Один из разработчиков языка Марк Миллер (Mark S. Miller) написал, что JavaScript «имеет фундаментальные изъяны» («Javascript has fundamental flaws…»), которые невозможно исправить. Поэтому и был создан *Dart*.

Первая общедоступная информация об этом языке программирования появилась 12 сентября 2011 года на конференции разработчиков Goto. 10 октября 2011 была проведена официальная презентация языка Google Dart.

Задачи, поставленные перед разработчиками языка:

* Создать структурированный и в то же время гибкий язык для [веб-программирования](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* Сделать язык похожим на существующие для упрощения обучения
* Высокая производительность получаемых программ как в браузерах, так и в иных окружениях, начиная от [смартфонов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD) и заканчивая серверами.

# Приложение

### Главная страница

На данный момент приложение состоит из двух страниц – стартовой (главной) и страницы группы с непосредственной навигацией между ними.

На главной странице пользователь в строке поиска может указать номер необходимой ему группы, затем отправляется асинхронный запрос к нам на сервер, приложение получает ответ в формате JSON и отображает перечень найденных по запросу групп.

Ниже будет приведена функция, формирующая и обрабатывающая запрос к API сервера.

|  |
| --- |
| Листинг 1.1. Отправка и обработка запроса |
| **void** \_findGroups(String group) **async** {  **results**.clear();  **final** String url = **"https://polytable.ru/search.php?query="**;  String query = url + group.trim();  **await** http.get(query)  .then((response) =>  json.decode(utf8.decode(response.**bodyBytes**))  .forEach((element) => **results**.add(**new** SearchResult(  name: element[**'name'**],  faculty\_abbr: element[**'faculty\_abbr'**]))))  .catchError((e) => print(e)); |

Далее приведём код виджета, в котором находится строка поиска и результаты поиска.

|  |
| --- |
| Листинг 1.2. Строка поиска и его результатов |
| ListView(  physics: **const** NeverScrollableScrollPhysics(),  children: <Widget>[  Padding(  padding:  **const** EdgeInsets.only(top: 60.0, right: 30.0, left: 30.0),  child: **new** TextField(  onSubmitted: \_findGroups,  style: TextStyle(  color: Colors.*black*, fontWeight: FontWeight.*bold*),  decoration: InputDecoration(  fillColor: Colors.*white*,  filled: **true**,  hintText: **"Группа"**,  border: OutlineInputBorder(  borderSide: **new** BorderSide(  color: Color.fromRGBO(16, 93, 59, 1.0))),  ),  ),  ),  (**results**.**isNotEmpty**)  ? Column(children: **results**)  : Container(width: 0.0, height: 0.0),  ], ) |

### Страница группы

На странице группы размещено расписание, полученное путём асинхронного запроса к тому же серверу и последующего ответа от него, а также календарь для удобной навигации между днями. Навигация между днями доступна как с помощью календаря, так и с помощью жестов «вправо» и «влево».

Недели – PageView, которые можно бесконечно пролистывать вправо и влево.

Каждый день – GridView, который можно пролистывать вниз и вверх, если не все пары уместились в экран. А так же, в зависимости от ориентации экрана число столбцов либо 1, либо 2.

Отображение каждой отдельно взятой пары реализуется классом Lesson, наследующим класс StatefullWidget, в котором хранятся название, тип и прочие данные о занятии. Данный класс содержит всего один метод, который строит по всем имеющимся данным непосредственно страницу.

|  |
| --- |
| Листинг 2.1. Класс Lesson |
| **class** Lesson **extends** StatefulWidget {  **const** Lesson(  {**this**.**title**,  **this**.**type**,  **this**.**time\_start**,  **this**.**time\_end**,  **this**.**teachers**,  **this**.**places**});   **final** String **title**;  **final** String **type**;  **final** String **time\_start**;  **final** String **time\_end**;  **final** List<**dynamic**> **teachers**;  **final** List<**dynamic**> **places**;   @override  \_LessonState createState() => **new** \_LessonState(); } |

Для удобной навигации между неделями и месяцами был написан виджет Calendar, в котором можно выбрать нужный день недели, перемещаться между неделями жестами «вправо» и «влево». Также можно вызвать полноценный Material Design календарь, нажав на соответствующую иконку.

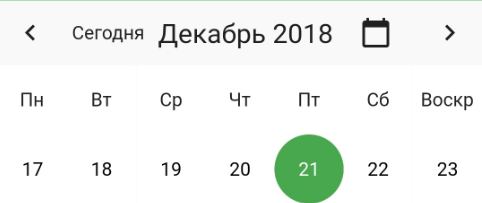


Рис.3. Календарь на неделю

### Обращение к Java - методу

Особенность реализации API Web-приложения, а именно, необходимость знать чётная или нечётная неделя для построения её расписания по двум имеющимся неделям, вынудила нас реализовать или использовать метод для получения номера недели в году.

Поскольку Flutter не предоставляет такой метод, было решено использовать так называемый канал между Flutter и Java. В классе MainActivity.java был написан метод, возвращающий номер недели в году.

|  |
| --- |
| Листинг 3.1. getWeekNumber (Java) |
| **private int** getWeekNumber(String date) {  SimpleDateFormat format = **new** SimpleDateFormat(**"yyyy-MM-dd"**);  **try** {  Date d = format.parse(date);   Calendar c = Calendar.*getInstance*();  c.setTime(d);   **return** c.get(Calendar.***WEEK\_OF\_YEAR***);  } **catch** (Exception e) {  **return** -1;  } } |

|  |
| --- |
| Листинг 3.2. Реализация канала (Java) |
| **new** MethodChannel(getFlutterView(), ***CHANNEL***).setMethodCallHandler(  **new** MethodCallHandler() {  @Override  **public void** onMethodCall(MethodCall call, Result result) {  **if** (call.**method**.equals(**"getWeekNumber"**)) {  **int** weekNumber = getWeekNumber(call.argument(**"date"**).toString());  System.***out***.println(weekNumber);  **if** (weekNumber == -1)  result.error(**"ERROR"**, **"Wrong data format present"**, **null**);  result.success(weekNumber);  } **else** {  result.notImplemented();  }  }  } ); |

|  |
| --- |
| Листинг 3.3. Функция, обращающаяся к каналу (Dart) |
| **static** Future<int> *getWeek*(String date) **async** {  int weekday = -1;   **await** *\_channel*.invokeMethod(**"getWeekNumber"**, <String, String>{  **"date"** : date  }).then((v) => weekday = v);   **return** weekday; } |

|  |
| --- |
| Листинг 3.4. Непосредственное использование написанных методов |
| bool isOdd = (**await** *getWeek*(key) - **staticStartWeek**) % 2 == 0; |

# Интерфейс приложения

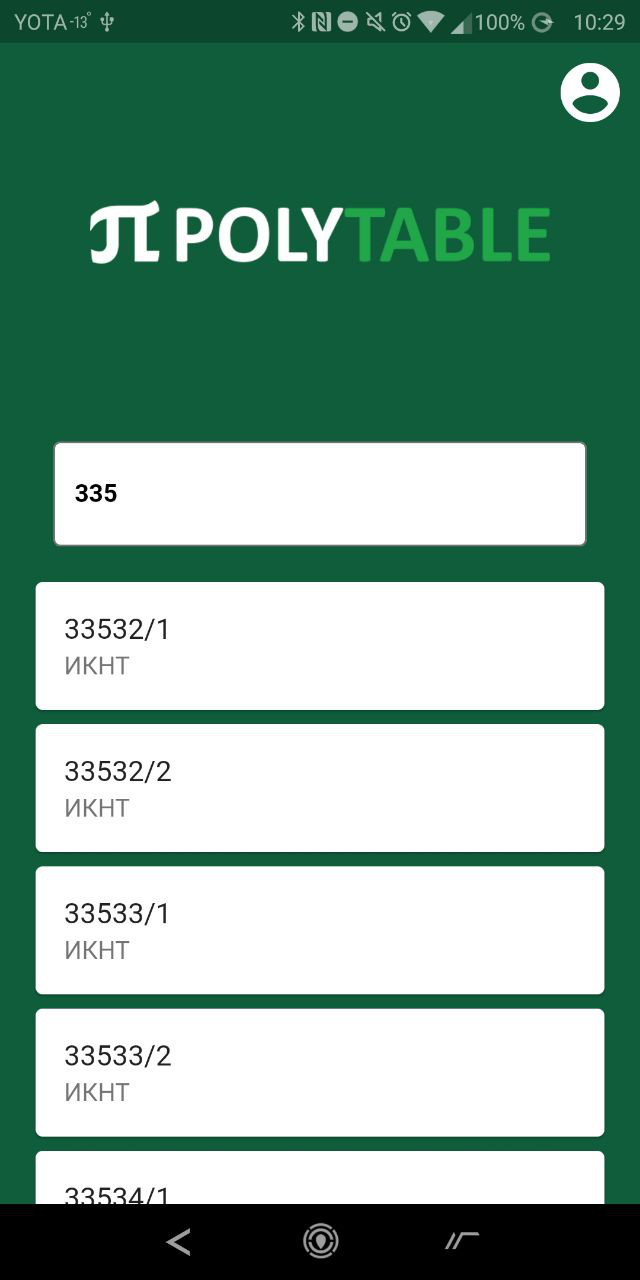
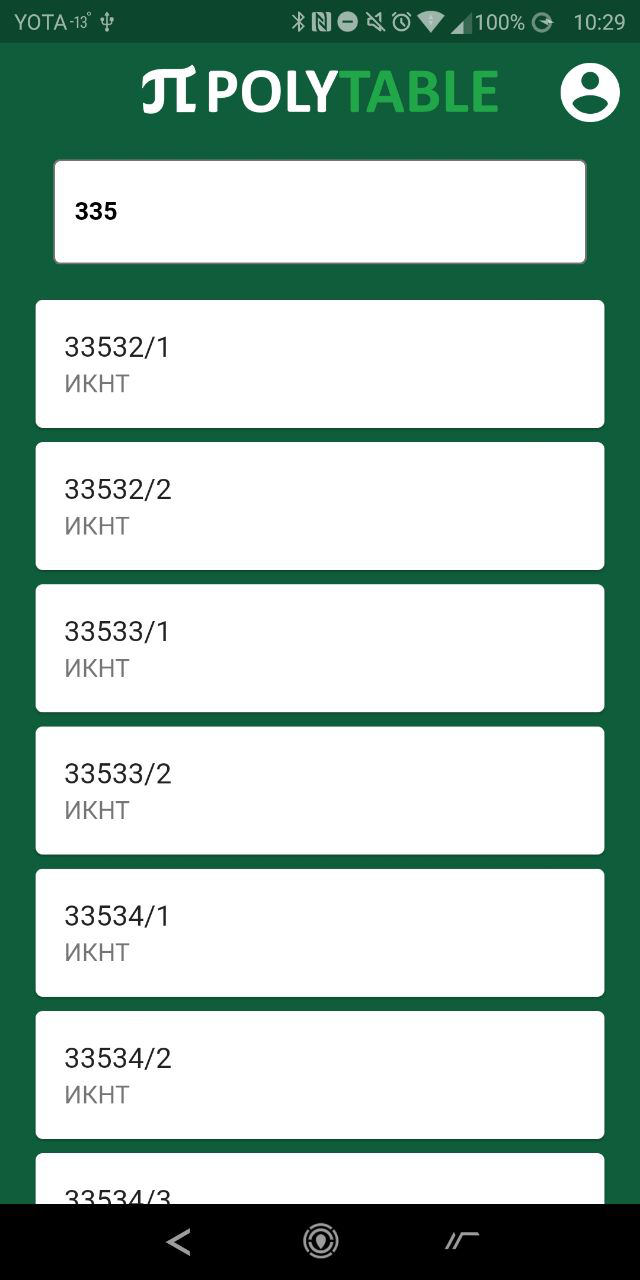
   

Рис.4. Главная страница

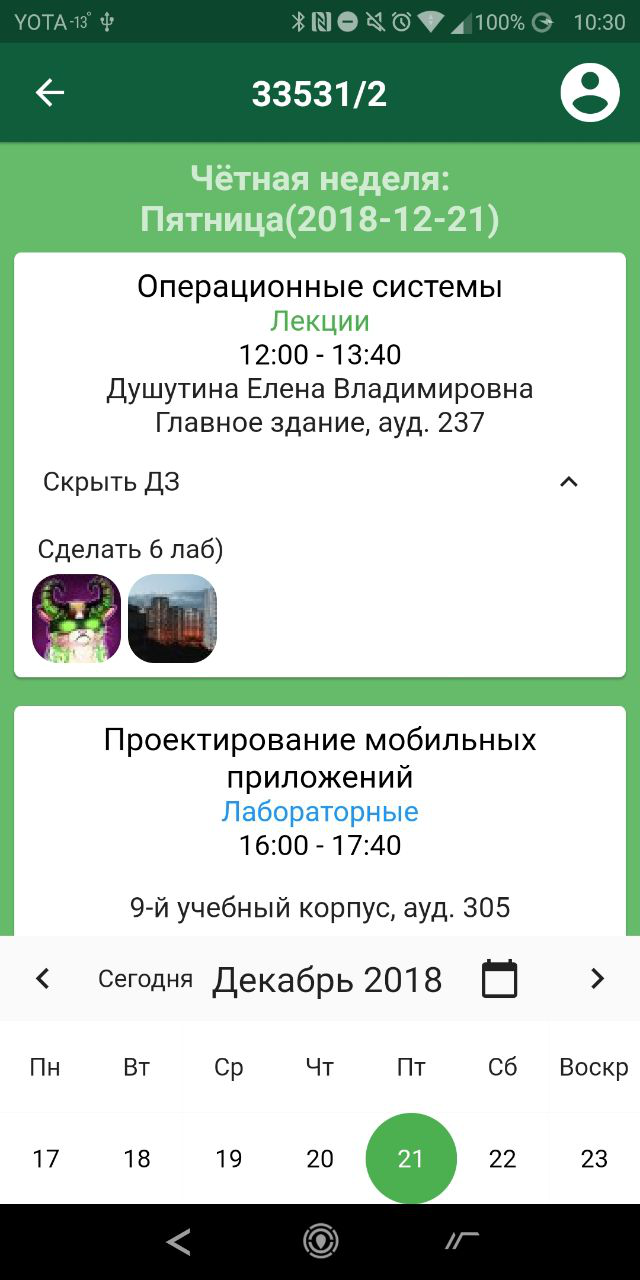
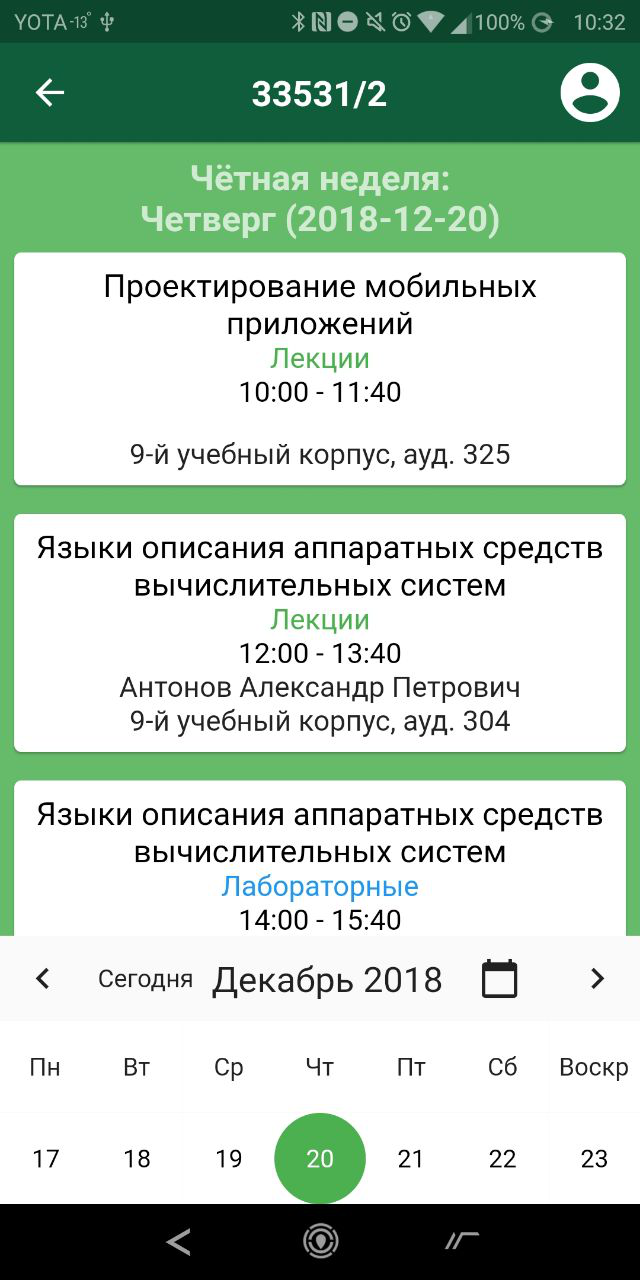
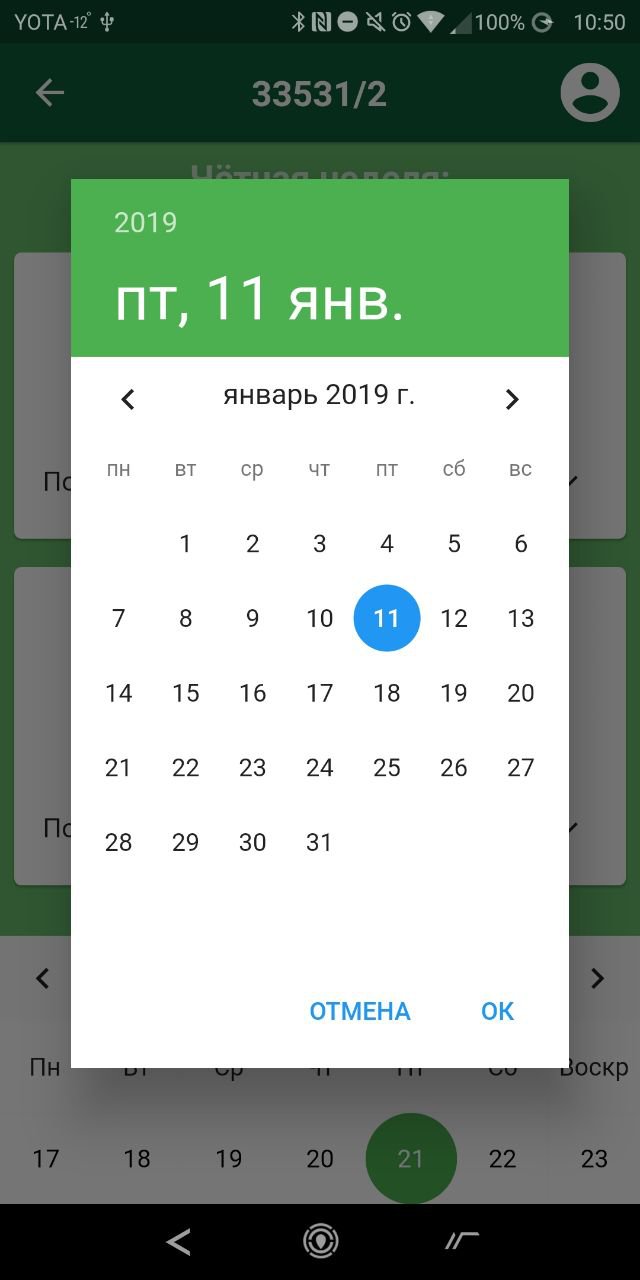
   

Рис.5. Страница группы

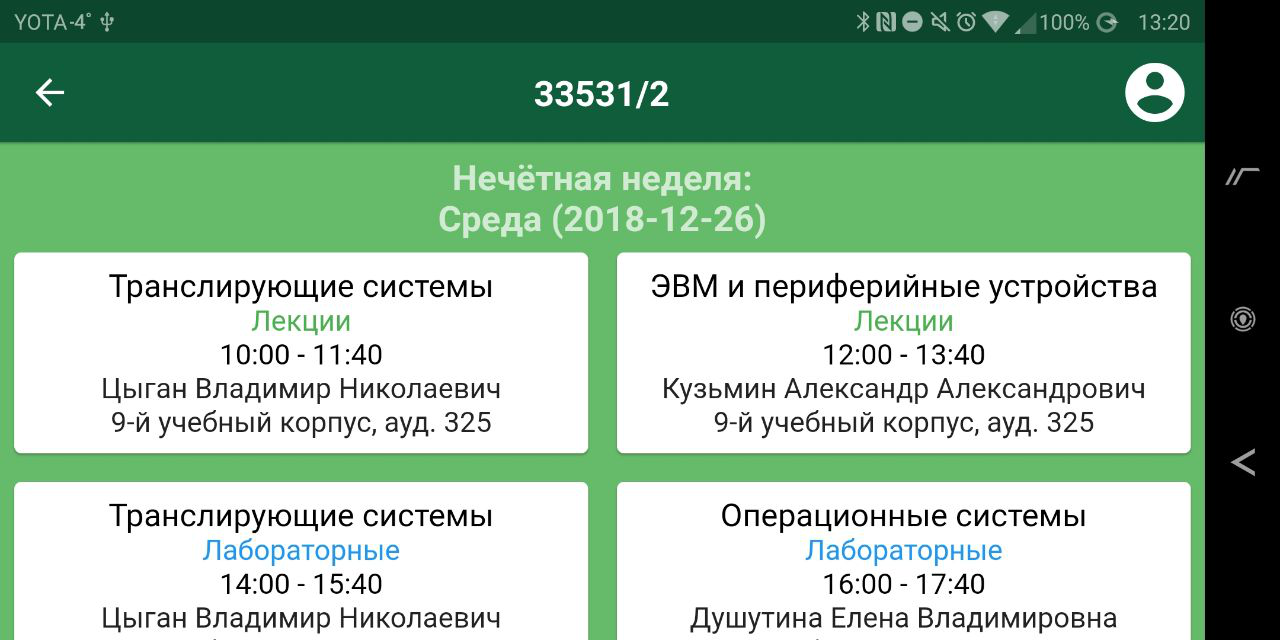


Рис. 6. Горизонтальная ориентация

# Тестирование и отладка

## Flutter analyze

Существует большое разнообразие инструментов и функций, помогающих отлаживать приложения Flutter. Перед запуском приложений протестируем код с помощью **flutter analyze**. Этот инструмент (который является оболочкой для инструмента dartanalyzer) анализирует код и помогает найти возможные ошибки. Он также встроен в Andoid Studio и не всегда необходимо вызывать его через консоль, достаточно обратить внимание на то, какие выражения IDE подчёркивает.

|  |
| --- |
| C:\Users\User\AndroidStudioProjects\polytable>flutter analyze  Analyzing polytable...  info - This class inherits from a class marked as @immutable, and therefore should be immutable (all instance fields must be final) - lib\calendar\flutter\_calendar.dart:10:7 -  must\_be\_immutable  info - Name non-constant identifiers using lowerCamelCase - lib\data\CalendarData.dart:8:23 - non\_constant\_identifier\_names  info - Name non-constant identifiers using lowerCamelCase - lib\templates\Lesson.dart:9:12 - non\_constant\_identifier\_names  info - Name non-constant identifiers using lowerCamelCase - lib\templates\Lesson.dart:10:12 - non\_constant\_identifier\_names  info - Name non-constant identifiers using lowerCamelCase - lib\templates\Lesson.dart:19:16 - non\_constant\_identifier\_names  info - Name non-constant identifiers using lowerCamelCase - lib\templates\Lesson.dart:20:16 - non\_constant\_identifier\_names  info - Name non-constant identifiers using lowerCamelCase - lib\templates\SearchResult.dart:5:39 - non\_constant\_identifier\_names  info - Name non-constant identifiers using lowerCamelCase - lib\templates\SearchResult.dart:8:16 - non\_constant\_identifier\_names  8 issues found. (ran in 3.1s) |

Первое замечание говорит о том, что все поля внутри flutter\_calendar должны быть final. А у нас одно поле – не final. Если сделать его final, то привязка дней к календарю будет работать некорректно.

Остальные замечания просят нас, чтобы мы следовали lowerCamelCase в названиях переменных. Но это замечание не является релевантным, т.к. названия дублируют названия полей из API Web-приложения.

## Тесты

Для тестирования приложения были написаны интеграционные и UI-тесты.

Интеграционные тесты содержатся в директории test. UI-тесты - в test\_driver.

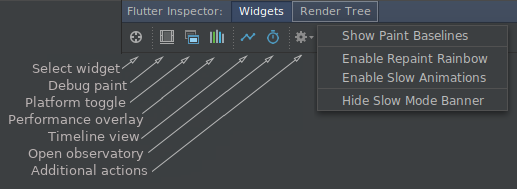
Суть интеграционных тестов заключается в создании лекции с размещённым внутри домашним заданием и последующей проверки его содержимого.

|  |
| --- |
| Листинг 4.1. Запуск интеграционных тестов |
| C:\Users\User\AndroidStudioProjects\polytable>flutter test  00:04 +2: All tests passed! |

UI-тест проверяет корректность работы Navigator’а, а именно – переход от главной страницы к страницы группы.

|  |
| --- |
| Листинг 4.2. Запуск UI-тестов |
| C:\Users\User\AndroidStudioProjects\polytable>flutter drive --target=test\_driver/UI.dart  Using device LLD L31.  Starting application: test\_driver/UI.dart  Initializing gradle... 1,1s  Resolving dependencies... 3,1s  Installing build\app\outputs\apk\app.apk... 13,1s  Gradle task 'assembleDebug'...  Gradle task 'assembleDebug'... Done 3,1s  Built build\app\outputs\apk\debug\app-debug.apk.  I/flutter (27637): Observatory listening on http://127.0.0.1:44386/  00:00 +0: main page tests (setUpAll)  [info ] FlutterDriver: Connecting to Flutter application at http://127.0.0.1:13669/  [trace] FlutterDriver: Isolate found with number: 748402500  [trace] FlutterDriver: Isolate is paused at start.  [trace] FlutterDriver: Attempting to resume isolate  [trace] FlutterDriver: Waiting for service extension  [info ] FlutterDriver: Connected to Flutter application.  00:02 +0: main page tests page load  00:04 +1: main page tests go to groups  00:05 +2: main page tests groups load  I/flutter (27637): Dynamic cache not present  00:08 +3: main page tests (tearDownAll)  00:08 +3: All tests passed!  Stopping application instance. |

Также при тестировании активно использовался Flutter Inspector, доступный в Android Studio. Ниже приведены его основные возможности.



*Рис.7. Возможности Flutter Inspector.*

# Вывод

В рамках данной курсовой работы было разработано приложение, представляющее из себя мобильную версию Web-приложения с использованием современных средств разработки, а именно библиотеки Flutter и языка программирования Dart. Были усвоены основы построения графических интерфейсов с использованием указанных средств. Были получены навыки разработки асинхронных приложений, взаимодействующие с интернет-ресурсами. На наглядном примере было рассмотрено взаимодействие Flutter и Andoid-платформы. Также были освоены методы Unit и интеграционного тестирования.

Ссылка на репозиторий: <https://github.com/grafovdenis/polytable>.

# Список литературы

1. [Электронный ресурс]: Пять причин проникнуться симпатией к Flutter

Режим доступа: <https://habr.com/company/ruvds/blog/349622/>

1. [Электронный ресурс]: Flutter Documentation  
   Режим доступа: <https://flutter.io/docs>
2. [Электронный ресурс]: TAGGED IN Flutter

Режим доступа: <https://medium.com/tag/flutter>