# **Лабораторная работа 11. Разработка мобильного приложения**

# Задачи лабораторной работы:

• Разработать собственное приложение с применением функционала, изученного в рамках курсах ПВМС.

• распределить работы между участниками группы и управлять проектом разработки, используя Github Projects.

• для контроля версий использовать git , разместив проект на github в репозитории для команды;

• документировать проект в Readme и wiki репозитория согласно требованиям, представленным ниже;

• непрерывная сборка проекта в Travis CI и автоматические тесты.

## Описание проекта.

Это приложение подойдет людям, записывающим или тем, кто только хочет начать записывать, все свои события. Имеетя возможность просматривать различные типы событий, загружаемые в приложение. Также имеется возможность добавить выбранное событие в избранное.

Имеется возможность просмотреть события, чтобы отпраздновать их. Также имеется возможность добавлять и оценивать свои собственные события. Вы можете установить это событие в дальнейшем уведомлении.

Имеется карта для указания мест с событиями.

## Распределение задач.

Роли проекта:

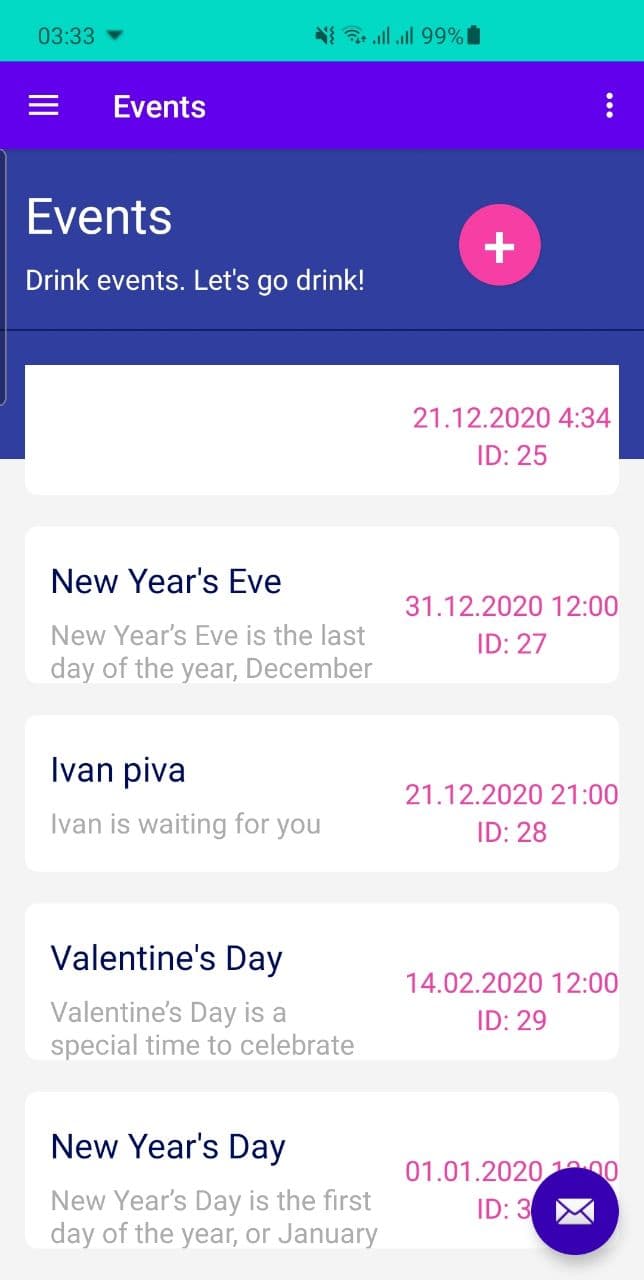
Булах Артём — backend/frontend, work with API.

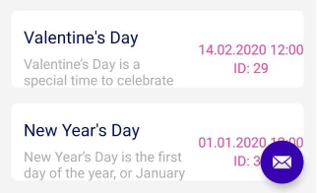
Руткевич Родион — backend/frontend , database, tests.

Козунов Алексей — backend/frontend, travis CI.

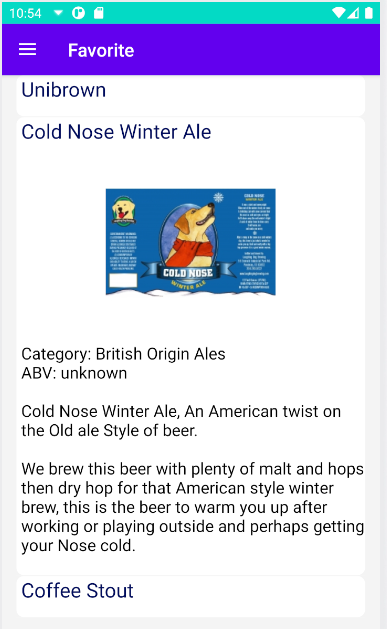
## Демонстрация работы

Главный экран:

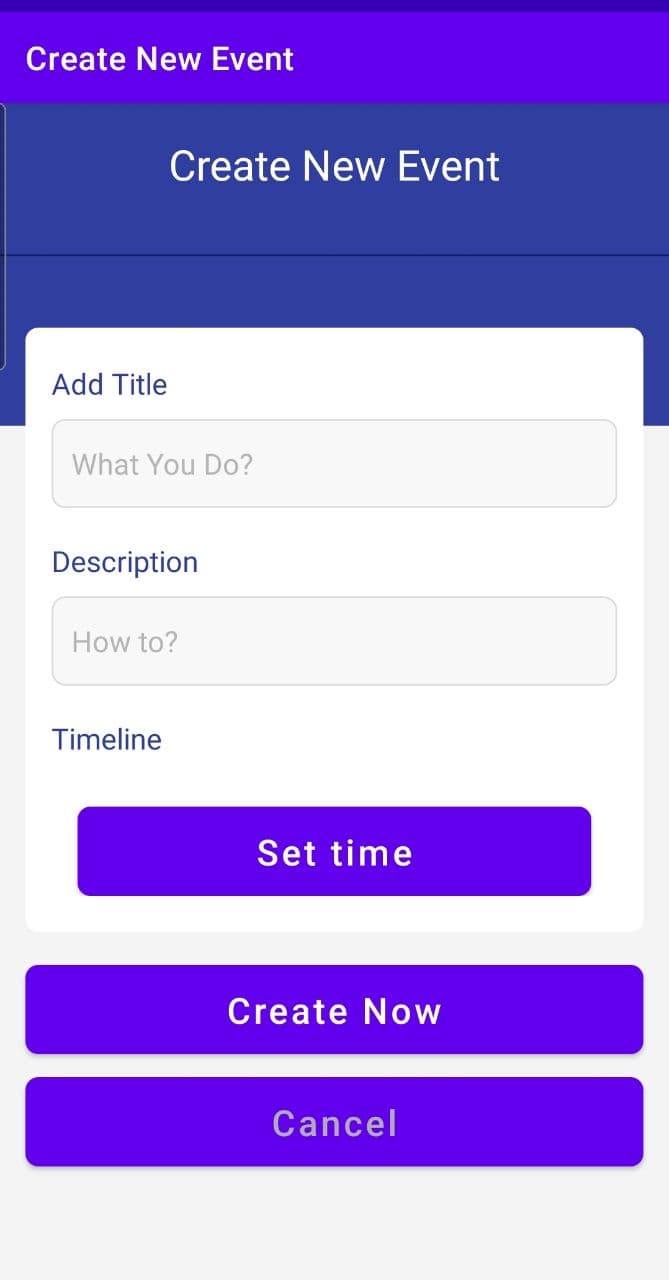


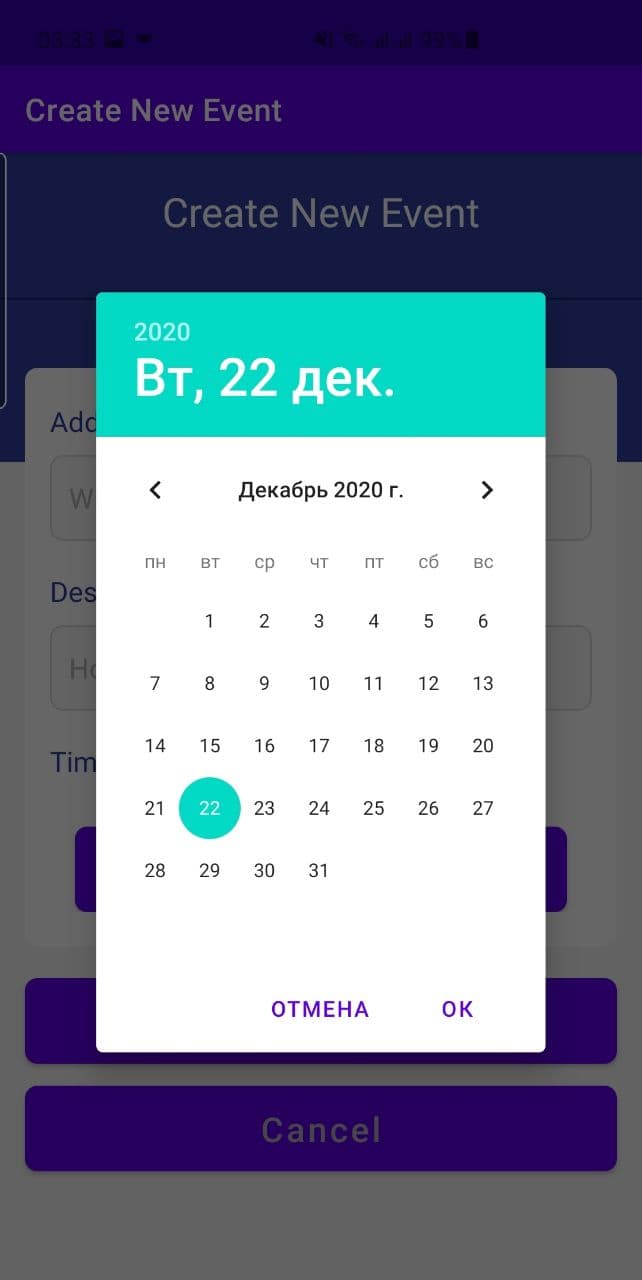


Каталог eventer’ов. Информация получается с помощью сервиса superevents.com.



Каталог сохраненных eventer’ов. Данные хранятся в db sqlite.

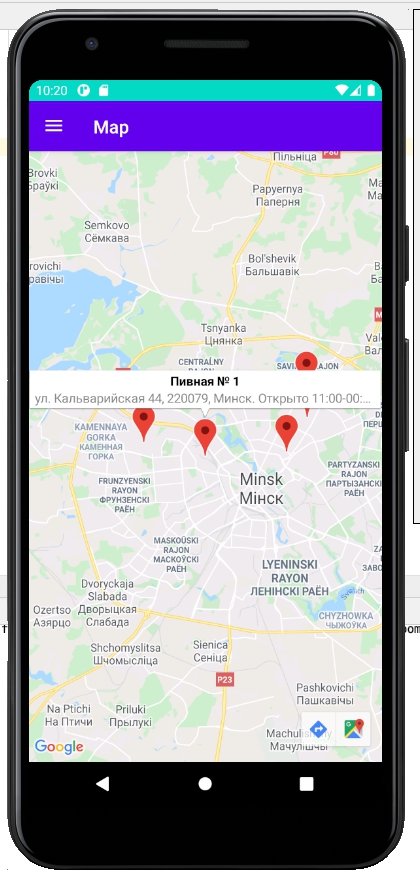




Редактор событый. События получаются с помощью сервиса calendarific.com.

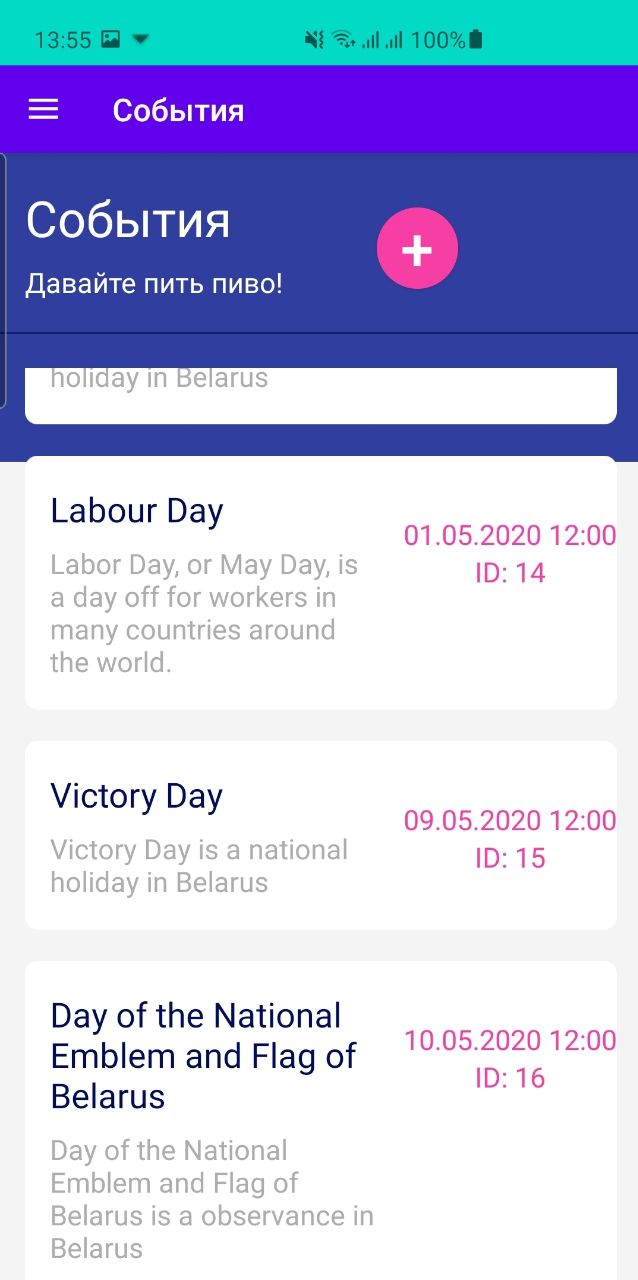
Так же можно добавлять свои собственные события.

Пользователь получает уведомления о ближайших событиях. Уведомления присылаются сервисом работающем в foreground.



Работы карты. Пользователь имеет возможность получить дпоплнителную информацию в каком-либо заведении.

В приложении поддерживается локализация на русском и английском языках.



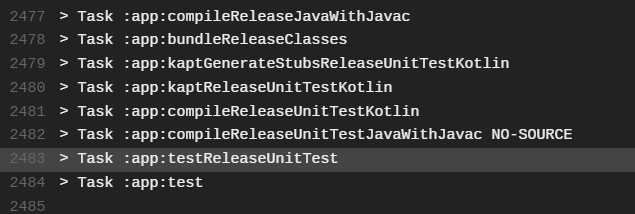
## Непрерывная сборка и тестирование.

Примеры тестов:

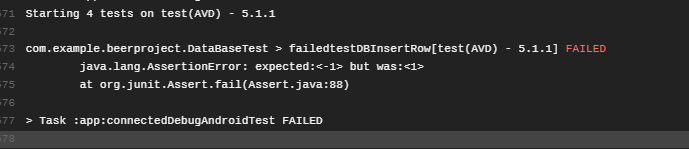
@Test  
fun testDBUpdateRow() {  
 // Context of the app under test.  
 val appContext = InstrumentationRegistry.getInstrumentation().*targetContext* // assertEquals("com.example.beerproject", appContext.packageName)  
 val dbHelper = DataBase(appContext)  
  
 val idRest = dbHelper.insertIntoEventTable("rest",  
 "bed", "31.12.2020 23:55")  
  
 assertEquals(1, dbHelper.updateDataInEventTable(idRest.toString(), "not\_rest",  
 "not\_bed", "25.12.2020 12:35"))  
  
 dbHelper.deleteRowFromEventTable(idRest.toString())  
}  
  
@Test  
fun testDBDeleteRow() {  
 val appContext = InstrumentationRegistry.getInstrumentation().*targetContext* val dbHelper = DataBase(appContext)  
  
 val idRest = dbHelper.insertIntoEventTable("rest",  
 "bed", "31.12.2020 23:55")  
  
 assertEquals(1, dbHelper.deleteRowFromEventTable(idRest.toString()))  
}

Для сборки и тестирования использовалась платформа travis ci.

Фрагмент работы с travis ci







Фрагмент .travis.yml

language: android

jdk: oraclejdk8

sudo: **false**

os: linux

android:

components:

- tools

- platform-tools

- build-tools-22.0.1

- android-22

- extra-google-m2repository

- extra-android-m2repository

- extra-android-support

# Specify at least one system image,

# if you need to run emulator(s) during your tests

- sys-img-armeabi-v7a-addon-google\_apis-google-22

- sys-img-armeabi-v7a-android-22

licenses:

- 'android-sdk-license-.+'

- 'google-gdk-license-.+'

- 'android-sdk-preview-license-.+'