1. 单元测试

单元测试是指将编写好的代码分成小部分来测试。我们通常编写具有类的程序，并将类中的逻辑进一步划分为方法。单元可以是方法、类或组件的一小段代码。单元测试的目的是验证各个单元的逻辑。

1. 编写简单的单元测试

使用 JUnit 框架来编写简单的单元测试。@Test 注解告诉 JUnit 它所附加的 public void 方法可以作为测试用例运行。要运行该方法，JUnit 首先构造一个新的类实例，然后调用注释方法。测试抛出的任何异常都会被报告被 JUnit 视为失败。如果没有抛出异常，则假定测试成功了。可以使用Google提供的Truth库来进行实际结果和预期结果的判断。

模拟是模拟或创建不真实行为的过程。在单元测试中，模拟有助于隔离外部依赖关系并创建预期事物的模拟行为。

1. 测试ViewModel

通常会在ViewModel的构造函数中传入对应的Repo，如果要模拟Repo，可以使用Mockito库。首先给测试类加上注解@RunWith(JUnit4::class)，有了这个 JUnit 将调用它引用的类来运行该类中的测试，而不是 JUnit 中内置的运行器。通过Mockito.mock(XX::class.java)来创建对应的模拟类。Mockito 的用法也可以根据条件使用 when() 和 thenReturn()。Mockito.when(boolean).thenReturn(xx())。

在编写测试时，我们需要在测试运行之前创建或执行一些东西，并在测试执行完成后清理它们，这是一个常见的要求。在某种程度上，它类似于 Viewmodels 的 init 和 onCleared 方法。我们使用注解@Before 和@After 来实现这一点。

如果要在单元测试中使用协程，需要增加kotlinx-coroutines-test库。

1. 测试Repo
2. 测试APIService

为了测试APIService，需要添加mockwebserver库。需要创建 MockWebServer 实例并将其与 Retrofit 一起使用以获得预期的结果。在这里，我们将响应数据设置为要在方法调用时返回的网络服务器。