Aplicación Android para visualización y ejecución de tests en Testlink.

Fco Javier Osuna Echavarría
GRADO DE INGENIERÍA INFORMÁTICA
Desarrollo aplicaciones dispositivos móviles (Android)

Nombre Consultor/a : Joan Vicent Orenga Serisuelo
Nombre Profesor/a responsable de la asignatura: Carles Garrigues Olivella

Enero de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aplicación Android para visualización y ejecución de tests en Testlink.*
Nombre del autor:	*Fco Javier Osuna Echavarría*
Nombre del consultor/a:	*Joan Vicent Orenga Serisuelo*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	*01 / 2017*
Titulación::	*GRADO DE INGENIERÍA INFORMÁTICA*
Área del Trabajo Final:	*Desarrollo aplicaciones dispositivos móviles (Android)*
Idioma del trabajo:	*English*
Palabras clave	*testlink*

Resumen del Trabajo:

La aplicación propuesta proporciona acceso a teslink desde un dispositivo Android, y lo hace en modo online y modo offline.
Proporciona la capacidad de ejecutar y visualizar documentos de planes de pruebas (test plans en adelante). La funcionalidad de edición no está incluída y se espera que los usuarios usen la interfaz de la instalación de testlink para ello.
La idea no es que la aplicación haga la función también de repositorio de resultados de ejecución de las pruebas, sino que los resultados se suban luego al servidor de testlink.
Gracias a que la aplicación puede funcionar en modo offline, sin acceso al servidor de teslink, es posible utilizar la aplicación en cualquier entorno y desde cualquier lugar. Para persistir toda la información usa una base de datos local.

Abstract:

The proposed application provides access to testlink from an Android device. It does it both in online and offline mode.
It provides test plan read and execution abilities, but no edit functionality, assuming that the users will rely on the testlink server installation for that.
Also tests results are meant to be uploaded to the testlink server once the testing is done.
The application should provide a more fluid and responsive UI than the legacy one provided by testlink.
Additionally, thanks to the offline mode, it will make possible to use the application

from everywhere, virtually in any environment.
In order to access testlink, it makes use of testlink xml-rpc API.
To preserve the downloaded data and the execution results. It relies on a local database.

Table of Contents

Introduction

Rationale and justification for the work

TestLink is a web-based test management system that offers support for test cases, test suites, test plans, test projects and user management, as well as various reports and statistics.

- Unfortunately, nowadays the UI is a bit outdated and there is no specific Android version: Partly the reason for that is that the UI is not rendered by means of recent technologies like nodeJS or angularJS, its a basic HTML that its generated by PHP scripts.
    - One of the drawbacks is that the HTML UI cannot handle hundreds (not to say thousands) of tests fluently, it scales poorly as the test case repository grows.
    - There are more modern commercial tools, some of them share the same issues, like Aptest, others provide better integration with control change tools, like the one provided by qTest.
- What really matters in this kind of tool is the way in which the information (test cases, test plans, test results) is structured, and in that regards, testlink is still valid, its a mature and well proven-software that has been widely used across multiple organizations and it provides the needed DB support that most software projects would need.

The intent of this TFG is to address some of testlink weak points (UI fluency) and also some functionality will be added that will make possible to extend the use of testlink to additional use cases (offline access). With an Android app for testlink it would be feasible to have:

- A more fluent UI than the original, that additionally would allow:
    - Download a set of testplans targeted for execution on an Android device.
    - Select one testplan targeted for execution, collect the results and push them back to a testlink server.
- Access from everywhere to a testplan, that would allow to extend the usage of the tool to other fields, for example, to test factory products, like a toaster appliance, or to review structures that are fixed, like a solar farm, or the mechanical parts of a damm.

- ◦ Thinking beyond, it would be nice to have video capture ability as well, to provide explicit feedback for failed test cases.

After a bit of research, it seems that currently there are no such tool:

> There are libraries that provide the java implementation of a testlink client (like https://code.google.com/archive/p/testlink-mob/)
> There is an HTML client specific for mobile devices (iOS and Android, but it seems that the project hasn´t fructified, since there is no submitted code). In any case, a thin HTML client could provide a convenient user experience, but wouldn't match the advantages of a native Android application, like offline access or UI fluency under low signal conditions.

Goals

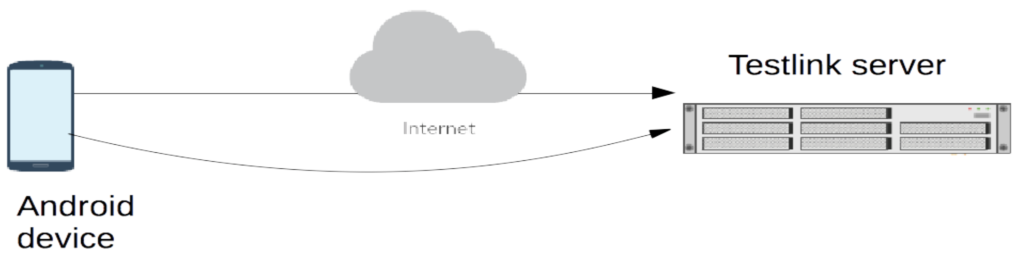
Design a native Android interface to access testlink:

- That allows offline access to test plans assigned to one single user. (Its not meant for shared use, one man, one Android device with the app).
- That allows offline read access to test descriptions contained in user’s test plans.
- That makes possible the execution of test plans assigned to that user and to upload the test results to testlink server (In places with connectivity)

The UI must be fluently enough (for example, making scrolling of testplans) and responsive (for example, submitting test results). Its expected to perform better than the original testlink UI in that regards.

Approach

The application will work in a semi-autonomous way but will depend on testlink for key functionality. Testlink will be accessed from the Android device either from a LAN or through the public internet



The development of the Android app will start from scratch, but making use of the remote API provided by testlink to access test cases and testplans and to submit the execution results.

We could consider specific products to integrate inside the application:

- To use apache axis2 for stub generation that provide the communication infrastructure with testlink.
- To persist the data retrieved from testlink (and the obtained test results) it will be considered to persist the info in a database.

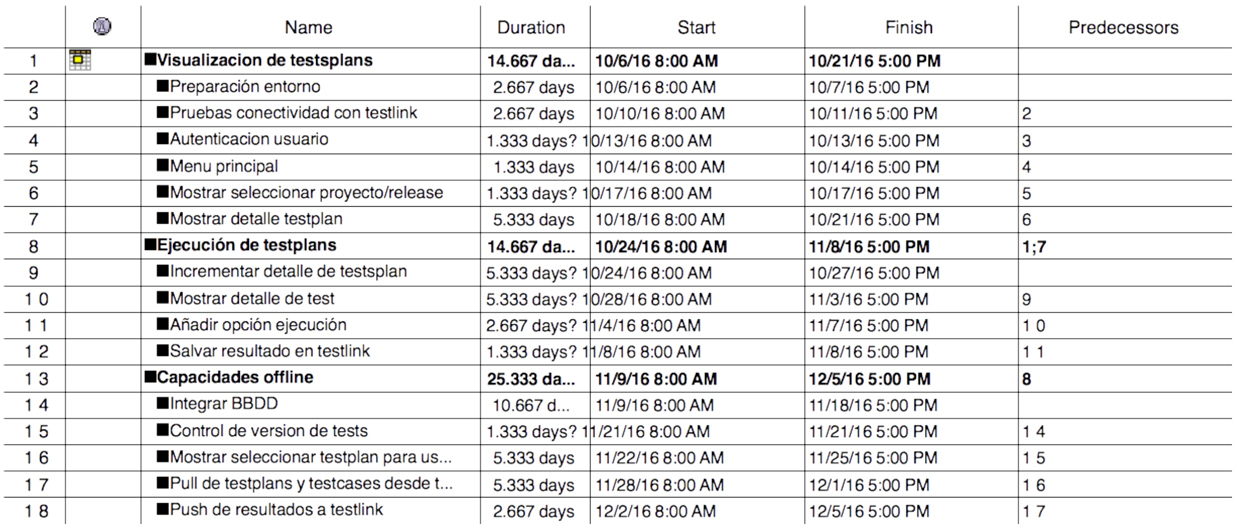
Planning

I plan to follow a spiral development model, in which I will be adding functionality and making changes to the original plan according to:

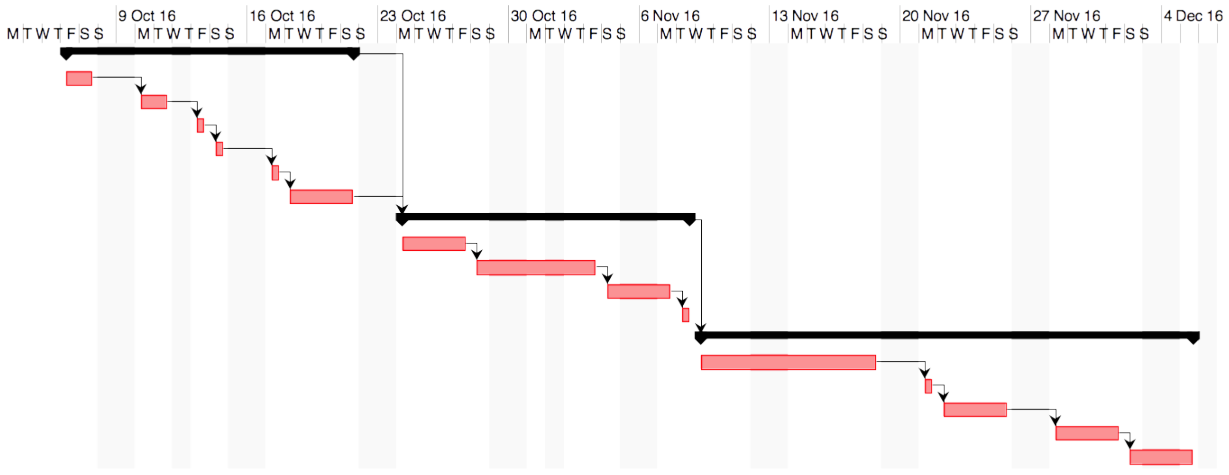
- Technical difficulties that probably I will be meeting.
- Design decisions/improvements that I will come up with as I move forward with the project.

The GANTT chart is included:

- *projectLibre* has been used to generate the chart. It allows you to set duration in days, but not in hours.
- The chart reflects only the planning for the product, it doesn't reflect the effort for documentation and partial deliveries.



		Name	Duration	Start	Finish	Predecessors
1		**Visualizacion de testsplans**	**14.667 da...**	**10/6/16 8:00 AM**	**10/21/16 5:00 PM**	
2		Preparación entorno	2.667 days	10/6/16 8:00 AM	10/7/16 5:00 PM	
3		Pruebas conectividad con testlink	2.667 days	10/10/16 8:00 AM	10/11/16 5:00 PM	2
4		Autenticacion usuario	1.333 days?	10/13/16 8:00 AM	10/13/16 5:00 PM	3
5		Menu principal	1.333 days	10/14/16 8:00 AM	10/14/16 5:00 PM	4
6		Mostrar seleccionar proyecto/release	1.333 days?	10/17/16 8:00 AM	10/17/16 5:00 PM	5
7		Mostrar detalle testplan	5.333 days	10/18/16 8:00 AM	10/21/16 5:00 PM	6
8		**Ejecución de testplans**	**14.667 da...**	**10/24/16 8:00 AM**	**11/8/16 5:00 PM**	**1;7**
9		Incrementar detalle de testsplan	5.333 days?	10/24/16 8:00 AM	10/27/16 5:00 PM	
10		Mostrar detalle de test	5.333 days?	10/28/16 8:00 AM	11/3/16 5:00 PM	9
11		Añadir opción ejecución	2.667 days?	11/4/16 8:00 AM	11/7/16 5:00 PM	10
12		Salvar resultado en testlink	1.333 days?	11/8/16 8:00 AM	11/8/16 5:00 PM	11
13		**Capacidades offline**	**25.333 da...**	**11/9/16 8:00 AM**	**12/5/16 5:00 PM**	**8**
14		Integrar BBDD	10.667 d...	11/9/16 8:00 AM	11/18/16 5:00 PM	
15		Control de version de tests	1.333 days?	11/21/16 8:00 AM	11/21/16 5:00 PM	14
16		Mostrar seleccionar testplan para us...	5.333 days	11/22/16 8:00 AM	11/25/16 5:00 PM	15
17		Pull de testplans y testcases desde t...	5.333 days	11/28/16 8:00 AM	12/1/16 5:00 PM	16
18		Push de resultados a testlink	2.667 days	12/2/16 8:00 AM	12/5/16 5:00 PM	17



Design

Users and context of use

Research methods

A mix of technics are going to be used:

Contextual observation/investigation

From my career experience, I had the opportunity to see that testlink is a common tool in QA departments, but there are some aspects of the tool that are not liked by the testers and even caused of complaint. The critics of the tools are directed against the UI, that looks outdated, its slow and the lack of integration with recent bug management tooks (like JIRA).

In the team meetings, my work colleges (QA), expressed their complaints regarding the usablity of the tool. There was a general opinion on the following points, corresponding to the weaknesses of the tool that could be improved:

- Lack of fluidity of the UI.
- Remote access too slow.
- The load times of the interface scale with the number of test cases for the projects:
    - When a project has more than 1000 test cases, latency largely compromises usability.

(I´m assuming that its an observation model, since it happens at the office, but since there is communication, then it would be interview too: contextual investigation)

One way to address these issues can be an Android application. The app would access a local copy of the data.

- Additionally, an Android app would free the testers main screen, so that they don´t have to switch the UI of the application that they are testing.
  - Doing it in this way, it wouldn´t be needed to switch frequently between the UI (or other testing tools) used to test the application subject of testing and the testlink UI.
- Specifically, to go over a list of test cases, the swipe gesture could be very intuitive.
- Additionally, because of the great availability of Android devices, one application allows us to move the tool to new use context, in which the tool hasn´t been used, for example in QA environments for manufacturing products (not IT).
  - Its possible to execute test plans without having a computer in front of you.
  - Its possible to execute test pans without network connection.

I will consider a new user profile to reflect it.

Regarding competitive analysis, in Google Playstore there aren´t similar applications. This might be part good, since there won´t be competitors, but it might indicate too that its not a good idea or there won´t be demand for the app. If that were the case, at least there would be some margin to tune the functionality to meet the demand (at least partly).

User profiles

Software testing engineer
• Professional testing engineer, between 20 and 65 years old (in practice its expected to be a person between 25 and 50 years old) ○ Testlink user, he is used to the tool (or even he masters it in some cases). ○ Its not assumed that he will have expertise on mobile apps, but because of his IT background, that wouldn't be an issue since he would learn it fast (at least till user level).

- User contexts:
  - From a IT workplace, at a desk with a computer, in an office.
  - From an Android device, using the device screen as auxiliary screen when his executing test plans from the computer´s screen as main screen.
  - During release verification periods, from a work desk and mainly (if not only) to execute functional testing.
- Tasks:
  - The user has previously created test plans in testlink (not the app, the server vesion), using the legacy testlink UI, and he uses the Android app for test plan reading and execution.
  - To configure the tool with his user and synchronize the data (pull the data from the server).
  - To choose the project/s in which the user is working and the specific build/releases.
  - To access the testplans from the application.
  - To open a test plan for execution, to execute the tests and submit the results.
- List of elements that must be present:
  - Synchronization and data connection forms.
  - To show the list of test plans for one specific project.
    - Ability to select one project, one testplan and a build.
  - Test cases corresponding to a test plan.
    - Test plan execution.
  - A side of the functional elements of the UI, that can be inspired in the legacy testlink HTML interface, we should focus on the non-functional aspects:
    - Fluent interface (in other case, the app has no meaning)
    - To keep focus on not to introduce any data synchronization issue between the app and the server side (testlink server).

Trying to broaden the application fields, this additional profile of specialist on quality assurance has been added.

- Its not specific on quality assurance, but on any product in general, manufactures that need to pass a quality control.
- Its a user profile that needs this Android application, but he hasn't realized yet: to use a manager that has many of the advantages comparing with keeping tests on paper or on an excel sheet.

Quality assurance specialist

- Specialist between 20 and 65 years old
  - Low computer skills: He doesn't know testlink and his Android knowledge is basic, probably he never installed an application.
- User contexts:
  - In a factory: in an appliance factory for example.
  - Each time that that a unit comes out of the assembly chain and its selected for quality check.
- Tasks:
  - The user is given a device (or the own one its configured for him) only with the set of test plans that he has to execute.
    - Test plans have been created forehead by another worker in testlink.
  - To select a test plan and execute it in offline mode:
    - Few test plans, only one project and release.
  - Upload the test plan results to testlink.
- List of elements that must be present:
  - Test plan list
  - Test case list of the test plan
  - For this user profile, the offline mode of the application is a significant feature.

Conceptual design

Personas:

María Fernandez 27 years old|QA engineer: she works in a company that develops a windows desktop application. She takes part of the validation for the upcoming release.

“I want to execute the tests and submit the results to testlink in a faster way than the legacy testlink interface allows me to do”

Goals:

- Execute the testplans targeted for the upcoming product release.
- I don´t want the limitations of the tool(testlink) to be an obstacle that brings me stress and extra time to complete my work.

Behaviors:

- Android device user

Expectations:

- An Android application that mitigates the weak points of testlink.

Juan López 55 years old | Quality control specialist: he work in an appliance factory, performing quality controls to wash machines

“I want to track the verifications that I did to a specific wash machine unit”

Low computer skills.

Goals:

- To be able to go over a comprehensive list of tests that I need to perform to a wash machine that comes out of the assembly chain.

- To have a record of the specific test that have been performed to a specific wash machine unit.

Behaviors:

• User with an Android device.

Expectations:

• To make easier the execution and to keep record of the validation results of one unit that comes out of the assembly chain.

Use scenarios

Note that María Fernández actually wants the functional part of the product, but this is already addressed in the legacy testlink itself: the way the information is arranged, project, releases, and test case fields.. the android application needs to implement a significant part of the functionality, but we have to focus on the use cases that are going to specific of the application:

- María Fernández wants to access testlink through a fluid interface.
- María Fernández wants the data both in the application and in the testlink server to match, without any synchronization issues.
- Juan López wants to show her manager that she performed a specific test on a specific was machine unit.
- Juan López wants to go over a check list to be executed on a wash machine unit that just came out of the production chain.

Additional use cases:

Further use cases can be identified. They are going to be targeted as optional, since it would require too much time so they are left as use case candidates for feature versions:

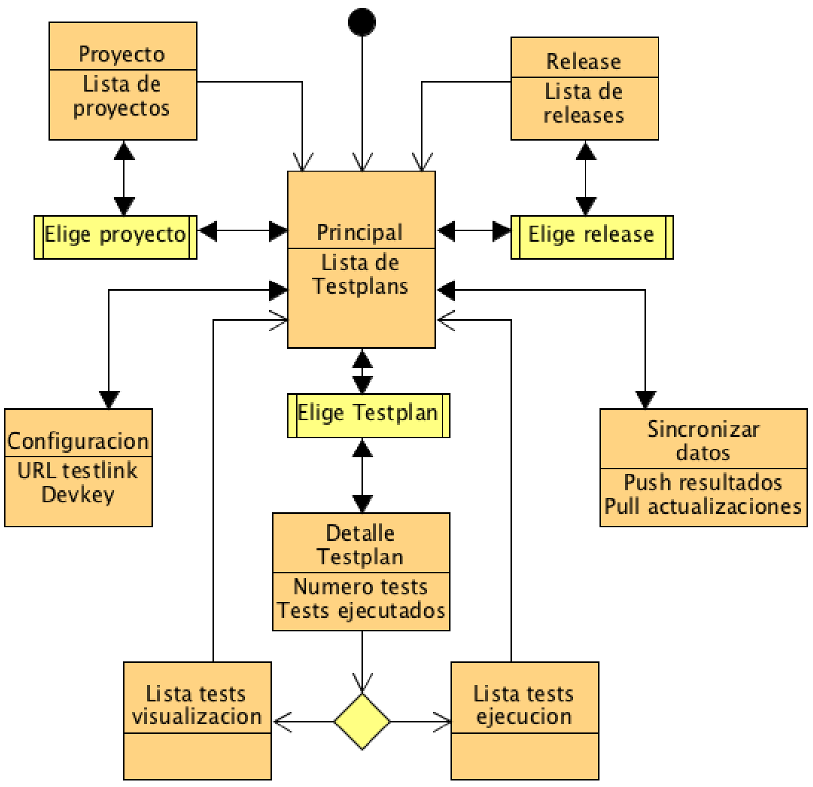
- María Fernandez wants to write new test cases directly in the application. That implies that the application must have edition abilities, and not only testcases/testplans read and execution.

- Juan López wants to record a video clip in case there is some failed tests, so that the engineer in charge of review the was machine doesn´t need detailed verbal or written description of the issue, that could be difficult.

Flowchart

The application starts showing a list of test plans. From here we have access to project and release filters: These filters help to keep the amount of test plan displayed under control. From the filter we can come back to the screen showing the test plan list to see filter already applied: only the test plans matching the filter criteria are displayed.

I we select one specific test plan, we move to a view in which the test plan details are rendered: total number of tests, failed tests, passed tests… .

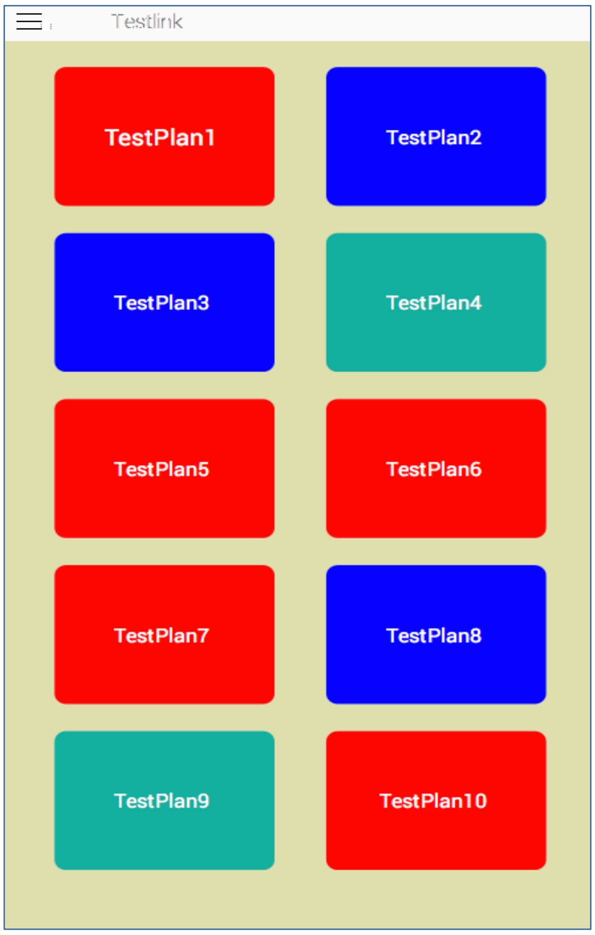


From the test plan list view, we have the option to move to the test plan details view. From test plan detail view, we can move to the test case set contained in the test plan for reading or move to the test case set for execution: In case we move to visualization, we will see the test case details (description, execution test steps…)and in case of execution, additionally there are controls to set one test case as passed or failed. We move from one test case to another by swiping on the screen.

Prototype

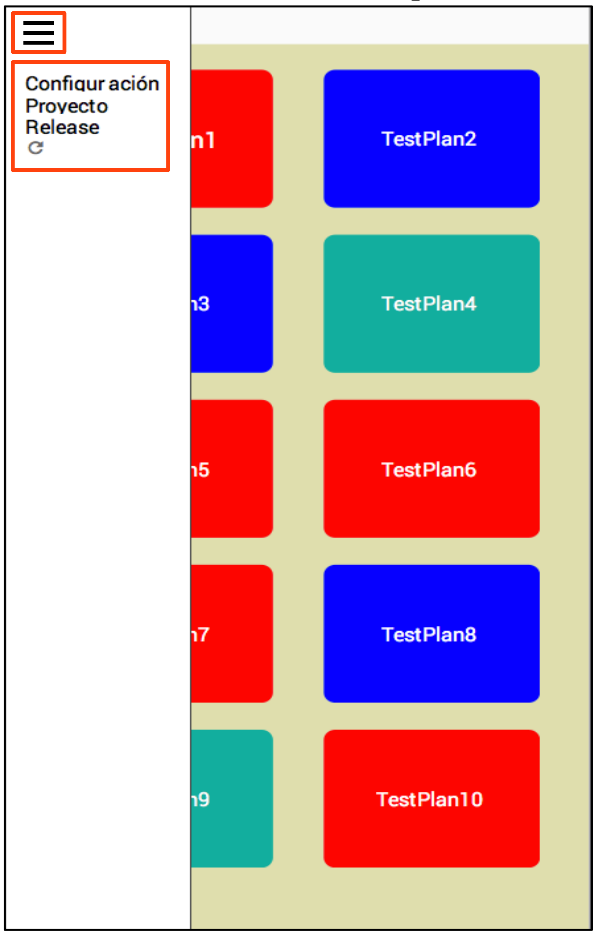
This is how the application will look like: (Omitting *material design* look and feel and conventions that I will try to follow. The mocks were generated using *Justinmind*)

Start screen



As we open the application, we see the testplans that we have locally stored. If we click on the burger icon (upper left corner) a slide menu is rendered.

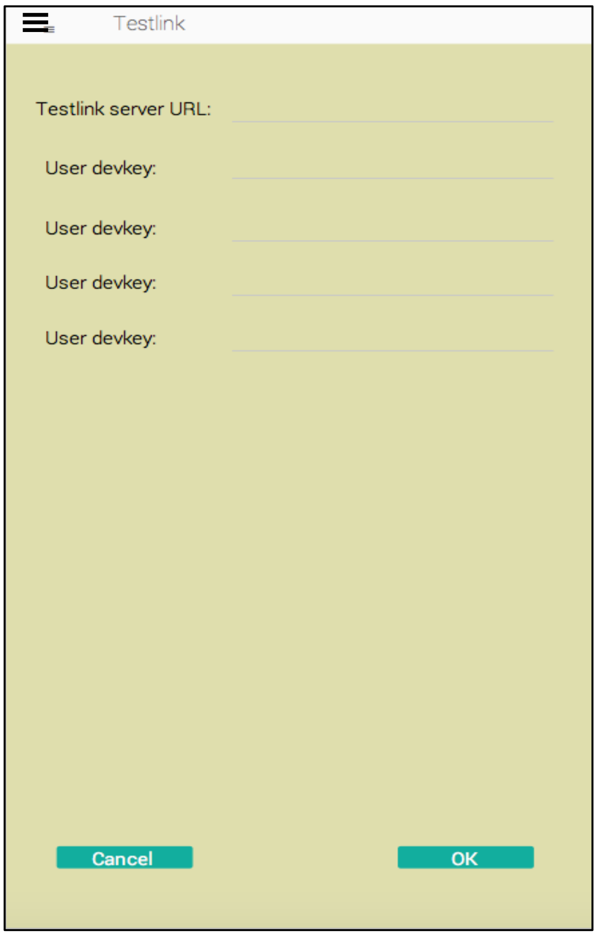
Slide menu to see the options



In the slide menu, we see the options (note that the burger and the options are highlighted in red)

- Access to configuration options
- Change project
- Change release
- Synchronize
  - Update testplan list
  - Upload test case execution results

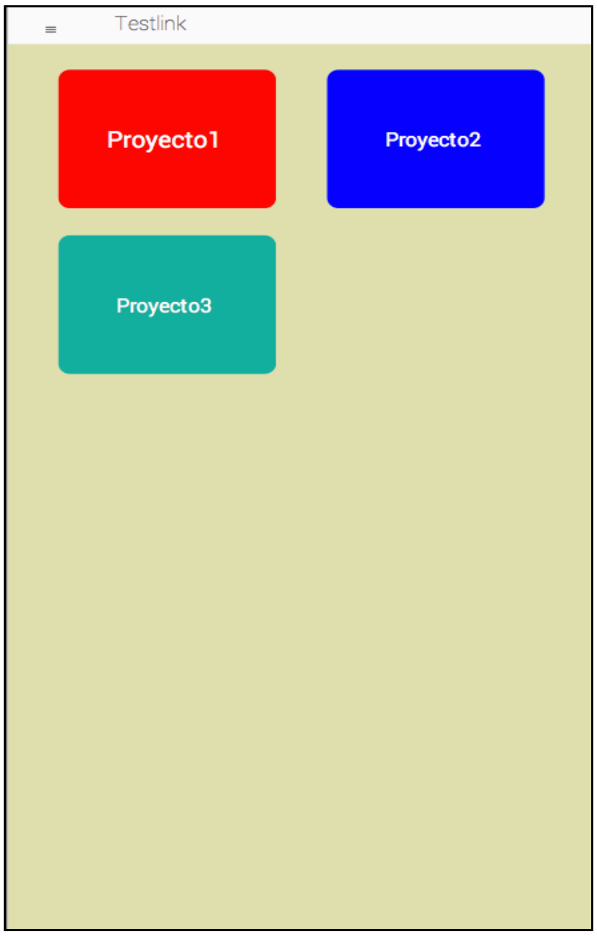
Configuration



In the configuration options, we can set:

- URL testlink of the testlink server
- User

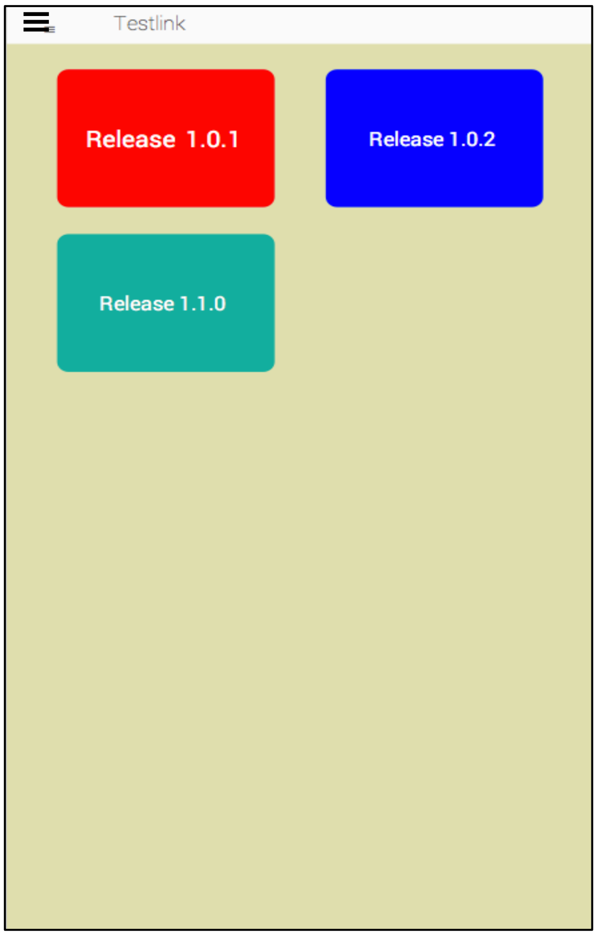
Select project/s



Here we see the projects that we want to retrieve their test plans and releases.

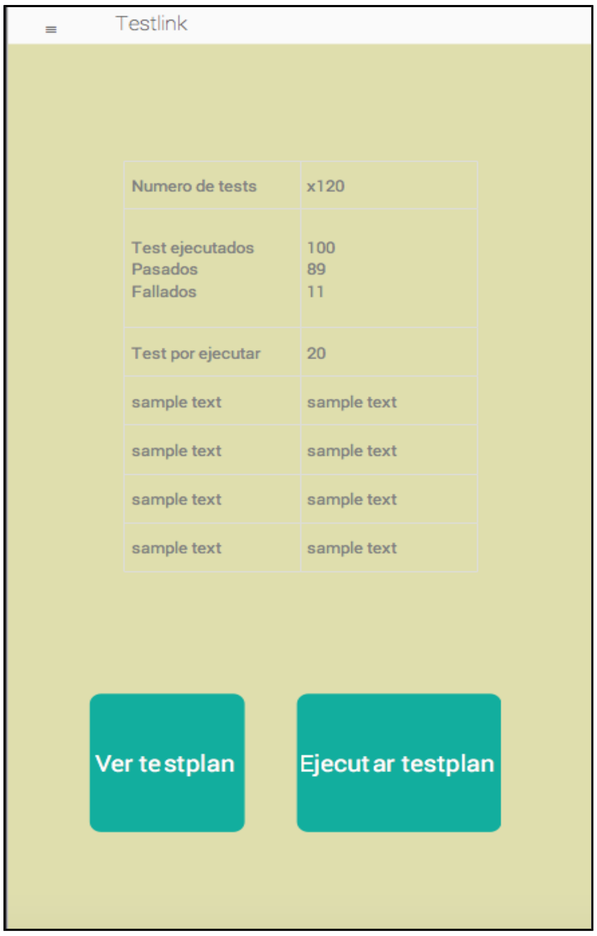
- We can select one or many
- The application saves the selection that we made. It is recovered among uses.
- To move back, we click on burger and we select test plans option

Choose release/s



- We see the projects whose test plans and releases we want to pull from testlink. We can select one or more
- The application saves the selection that we did among sessions.
- To go back, burger icon and test plan options.

Test plan details

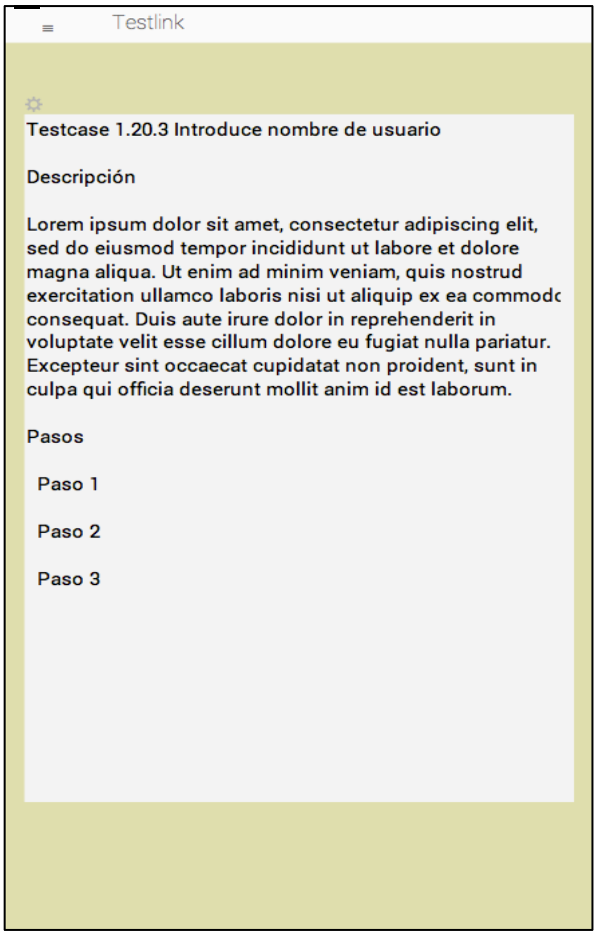


Numero de tests	x120
Test ejecutados Pasados Fallados	100 89 11
Test por ejecutar	20
sample text	sample text
sample text	sample text
sample text	sample text
sample text	sample text

When we click on a specific test plan, the test plan details are displayed, among others:

- Total number of test cases, number of executed test case, number of passed test cases..
- Additionally we have buttons to display the test plan contents and to execute it.

Test plan visualization


Testlink

Testcase 1.20.3 Introduce nombre de usuario

Descripción

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Pasos

Paso 1

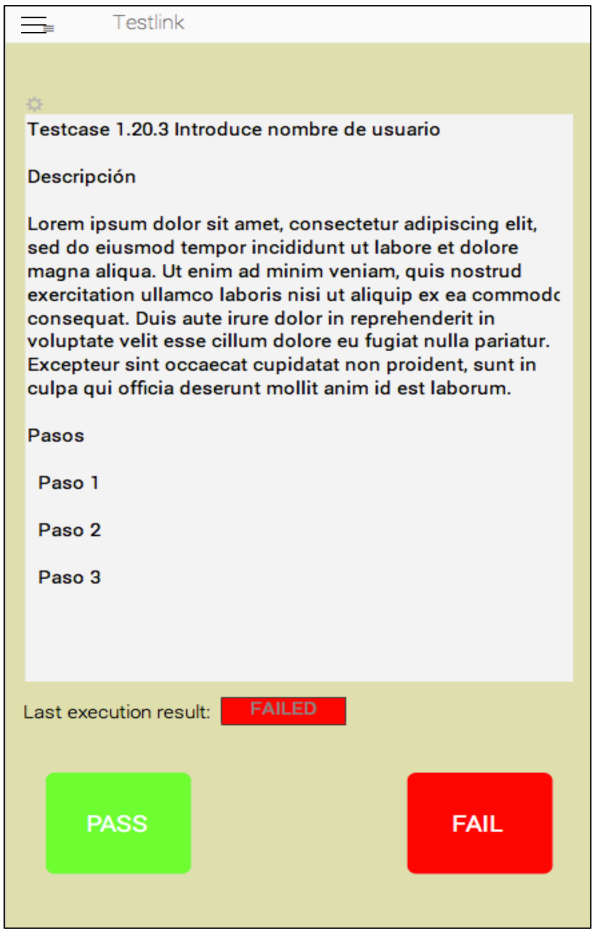
Paso 2

Paso 3

We move from one test case to another using the swipe gesture, either links or right.

- For each test case, we can read the details.

Test plan execution


Testlink

Testcase 1.20.3 Introduce nombre de usuario

Descripción

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Pasos

Paso 1

Paso 2

Paso 3

Last execution result: FAILED

- We move from one test case to another using the swipe gestures,. For each test case, we can read the details, among them, the last execution result.
- We click on the PASS button or on FAIL to record the execution result.

Evaluation

To perform the evaluation, we will select three users with **Testing engineer** profile and one with profile of **Quality Assurance specialist**.

While they are using the application, the following questions will be asked:

- Has it be intuitive to synchronize your test plans with the testlink server?
    - Have you experimented any issue to do it?
- Did you manage to select one project?
- Did you chose any release?
- Were you able to chose the testplan that you wanted?
- Did you manage to synchronize the results when you were done with the testing?
- Did you like the fluency of the UI?

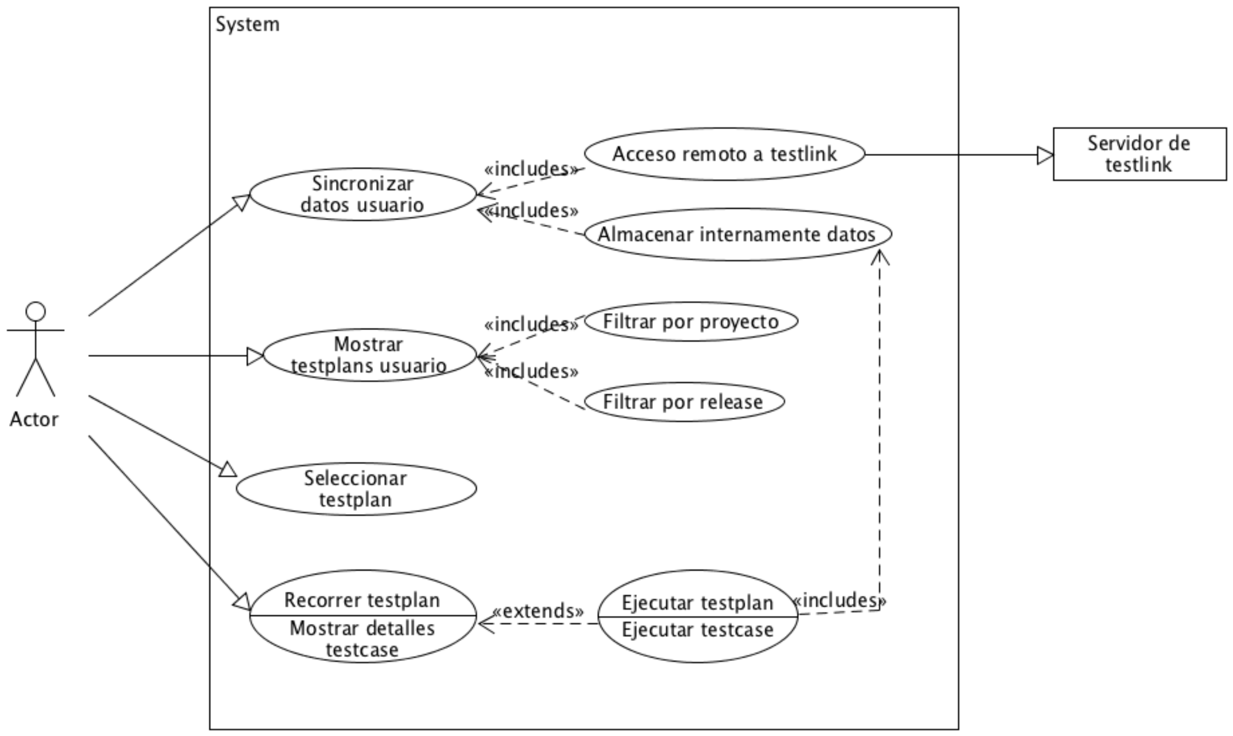
These are the tasks that the users must accompli:

- To configure the application with the testlink configuration data.
- Synchronize the application content with the testlink server.
- Select one project.
- Choose one release.
- Move through the test plan list.
- Select one specific test plan.
- Read the content of one test plan.
- Execute some of the test case contained in one test plan.
- Synchronize the results with the testlink server and verify that the test case execution results have been uploaded.

Questions around the tasks are in the previous paragraph and they will be asked the users before and after requesting them to perform such tasks.

Specification

Use cases



Id	CU-003
Name	**Synchronize user data**
Priority	High
Description	I want the application to synchronize its data with the testlink server: • Using the user details and the connection settings. • Both for data download (test plan and test cases) and data upload (test execution results). • Avoiding synchronization issues between the application and the server.
Actors	Testing engineer/Quality assurance specialist

Preconditions	• There is persistence mechanism in the application in which the data downloaded from testlink can be saved. CU-003 • There is a communication mechanism with testlink. CU-002
Started by	User
Flow	The user sets the connection data and synchronizes the application.
Postconditions	There are data synchronization issues between the application and the server.
Notes	

Id	CU-001
Name	**Testlink remote access**
Priorirty	High
Description	The application accesses programmatically the testlink server by means of a remote call.
Actors	Testing engineer/Quality assurance specialist
Preconditions	The connection settings: server details user credentials
Started by	The user as part of CU-003
Flow	The user sets the connection data and invokes a synchronization.
Postconditions	-
Notes	

Id	CU-002
Name	**Data persistence**
Priorirty	High
Description	• The testlink data that are displayed in the device are persisted locally. • The test case executing results are persisted locally. • The application can work.offline, without connection with the testlink server. • The user can synchronize on demand (its in that moment when the testlink server will be accessed).
Actors	Testing engineer/Quality assurance specialist
Preconditions	CU-001
Started by	The user as part of CU-003
Flow	The user sets the connection details and synchronizes the appplication.
Postconditions	The data between the testlink server and the application are aligned after completing the synchronization.
Notes	

Id	CU-004

Name	**Show test plans**
Priorirty	High
Description	• The application shows a list of test plans from the data persisted locally. • Moving through the test plan list provides a feeling of fluency.
Actors	Testing engineer/Quality assurance specialist
Preconditions	The testplans are persisted locally (CU-001)
Started by	User
Flow	The user moves through the plan list.
Postconditions	-
Notes	

Id	CU-005
Name	**Project filter**
Priorirty	High
Description	I want to select one project, the test plan list must show only the test plans that are created for that project. By means of a menu item its accessed a project filter that shows the different projects.
Actors	Testing engineer/Quality assurance specialist

Preconditions	CU-003 y CU-004
Started by	
Flow	The user opens the project filter, sets some condition and observes the effect applied on the test plan list.
Postconditions	The selected projects are saved as preferences.
Notes	

Id	CU-006
Name	**Release filter**
Priorirty	High
Description	I want to select one release, the test plan list must show only the test plans that are targeted for that release. By means of a menu item its accessed a release filter that shows the different releases.
Actors	Testing engineer/Quality assurance specialist
Preconditions	CU-003 y CU-004
Started by	User
Flow	The user opens the release filter, sets some condition and observes the effect applied on the test plan list.
Postconditions	The selected releases are saved as preferences.
Notes	

Id	CU-007
Name	**Select one test plan**
Priorirty	High
Description	From the test plan list (CU-004) a specific test plan is selected. The test plan details are rendered: • Project and release • Test plan name and description • Test case statistics = total, executed, passed, failed
Actors	Testing engineer/Quality assurance specialist
Preconditions	CU-004
Started by	User
Flow	The user selects one test plan.
Postconditions	-
Notes	

Id	CU-008

Name	**Test plan walk through**
Priorirty	High
Description	I want to go over the test case details and descriptions: • The move from one test case to another must be fluid, making use of the touch interface benefits (for example, swipe) • Of each test case, the needed info to understand the test must be rendered.
Actors	Testing engineer/Quality assurance specialist
Preconditions	CU-007
Started by	User
Flow	The user selects a test plan and from test plan details, the test plan visualization option is selected. The user moves through the test cases of the test plan.
Postconditions	-
Notes	It could show what was the last execution result for each test case.

Id	CU-009
Name	**Test plan execution**
Priorirty	High
Description	I want to go over the test case descriptions and details of the test case that are going to be executed:

	• The move from one test case to another must be fluid, making use of the touch interface benefits (for example, swipe) • Of each test case, the needed info to execute it must be rendered. • There are UI controls to pass or fail the test.
Actors	Testing engineer/Quality assurance specialist
Preconditions	CU-007
Started by	User
Flow	The user selects a test plan and from test plan details, the test plan execution option is selected. The user moves through the test cases of the test plan, executing them.
Postconditions	The test execution result is stored internally and uploaded to the test link server in the next data synchronization.
Notes	

Development

List of tools used:

Main tools

- Android studio SDK (2.2.3) as IDE.
- Virtual box (5.1.8)
  - It is used to run a testlink server instance (1.9.15)

Unfortunately it was not feasible to run the Android device emulator that is included with Android studio. The reason for that is that it cannot run in the same machine with Virtual box (both tools collide, some shared library mess).

Libraries used in the android project

Included dependencies as reported by gradle:

```
+--- org.apache.xmlrpc:xmlrpc-client:3.1.3
|    \--- org.apache.xmlrpc:xmlrpc-common:3.1.3
|         \--- org.apache.ws.commons.util:ws-commons-util:1.0.2
+--- com.android.support:appcompat-v7:25.0.1
|    +--- com.android.support:support-v4:25.0.1
|    |    +--- com.android.support:support-compat:25.0.1
|    |    |    \--- com.android.support:support-annotations:25.0.1
|    |    +--- com.android.support:support-media-compat:25.0.1
|    |    |    \--- com.android.support:support-compat:25.0.1 (*)
|    |    +--- com.android.support:support-core-utils:25.0.1
|    |    |    \--- com.android.support:support-compat:25.0.1 (*)
|    |    +--- com.android.support:support-core-ui:25.0.1
|    |    |    \--- com.android.support:support-compat:25.0.1 (*)
|    |    \--- com.android.support:support-fragment:25.0.1
|    |         +--- com.android.support:support-compat:25.0.1 (*)
|    |         +--- com.android.support:support-media-compat:25.0.1 (*)
|    |         +--- com.android.support:support-core-ui:25.0.1 (*)
|    |         \--- com.android.support:support-core-utils:25.0.1 (*)
|    +--- com.android.support:support-vector-drawable:25.0.1
|    |    \--- com.android.support:support-compat:25.0.1 (*)
|    \--- com.android.support:animated-vector-drawable:25.0.1
|         \--- com.android.support:support-vector-drawable:25.0.1 (*)
+--- com.android.support:support-v4:25.0.1 (*)
+--- org.apache.commons:commons-configuration2:2.1
|    +--- org.apache.commons:commons-lang3:3.3.2
|    \--- commons-logging:commons-logging:1.2
\--- com.android.support:design:25.0.1
     +--- com.android.support:support-v4:25.0.1 (*)
     +--- com.android.support:appcompat-v7:25.0.1 (*)
     +--- com.android.support:recyclerview-v7:25.0.1
     |    +--- com.android.support:support-annotations:25.0.1
     |    +--- com.android.support:support-compat:25.0.1 (*)
     |    \--- com.android.support:support-core-ui:25.0.1 (*)
```

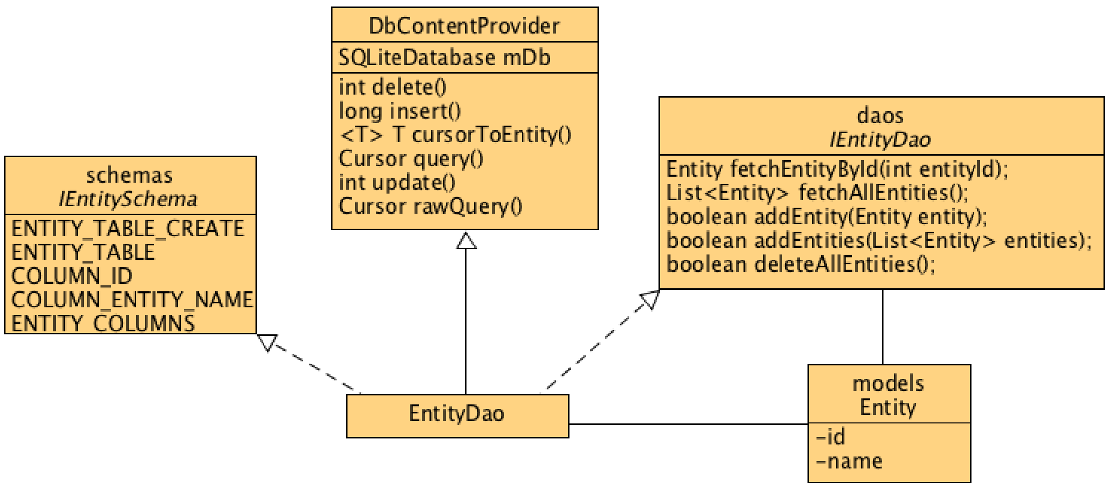
```
          \--- com.android.support:transition:25.0.1
               \--- com.android.support:support-v4:25.0.1 (*)
```

Some design decisions and implementation details

In order to access SQLlite database, the following database access pattern is used:

http://wale.oyediran.me/2015/04/02/android-sqlite-dao-design/

This is an example of database access to an *entity* (It doesn't correspond to any entity of the project, it´s just an example):



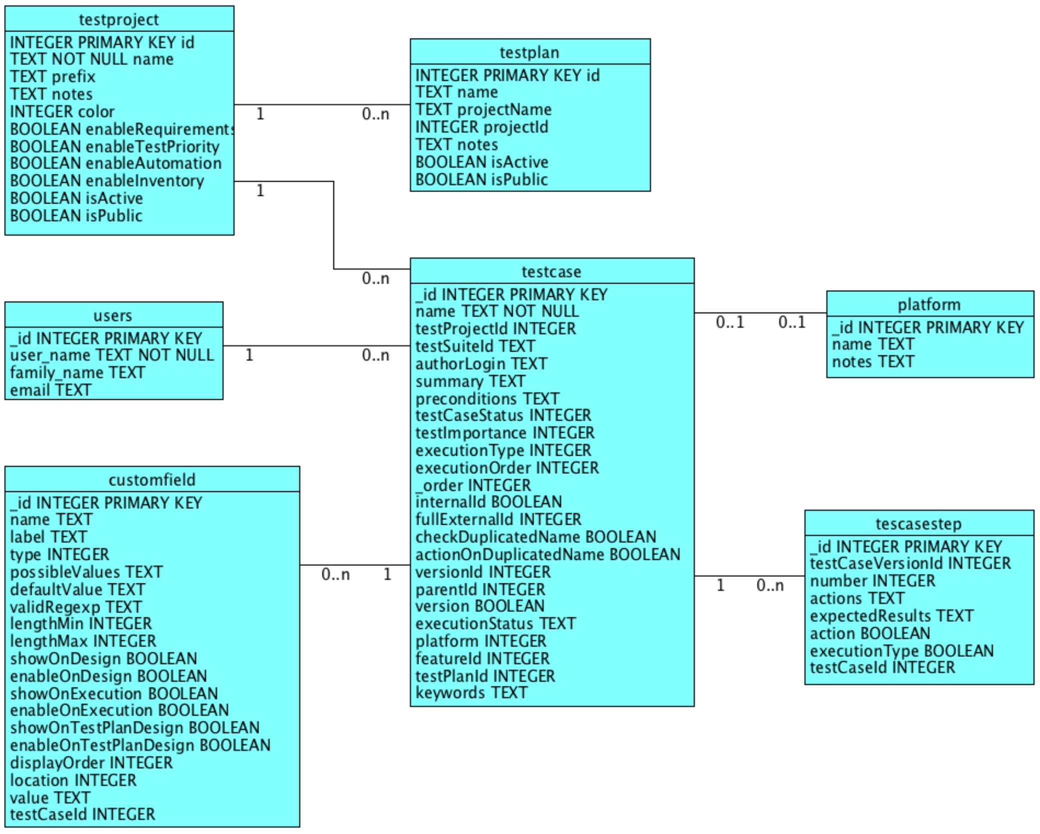
Brief explanation:

- It relies on the db that is included in the Android framework (*SQLite*)
- It shares the model implementation classes with the testlink access layer (that is based on *xml-rpc*, as described later on).
- The SQL table definition syntax is included in the **IEntitySchema** interface.
- **IEntityDao** contains the access methods that will be exposed by the data access object.
- **DbContentProvider** is the class that encapsulates the actual SQLite access:
  - Has a reference member to **SQLiteDatabase**
  - Provides the db usage methods (query/update/insert/delete)

- Finally, **EntityDao** provides the implementation of the methods that are exposed to the application.
  - Internally, these methods´ implementation is making use of the SQL syntax contained in **IEntitySchema** and the db access methods of **DbContentProvider**.

Later on, it´s mentioned that this layer testing will be automated by *intrumented* unit tests.

DB schema:



Note that I haven´t ported *testlink*´s *TestSuite*, since its a table oriented to group *TestCase* definitions.

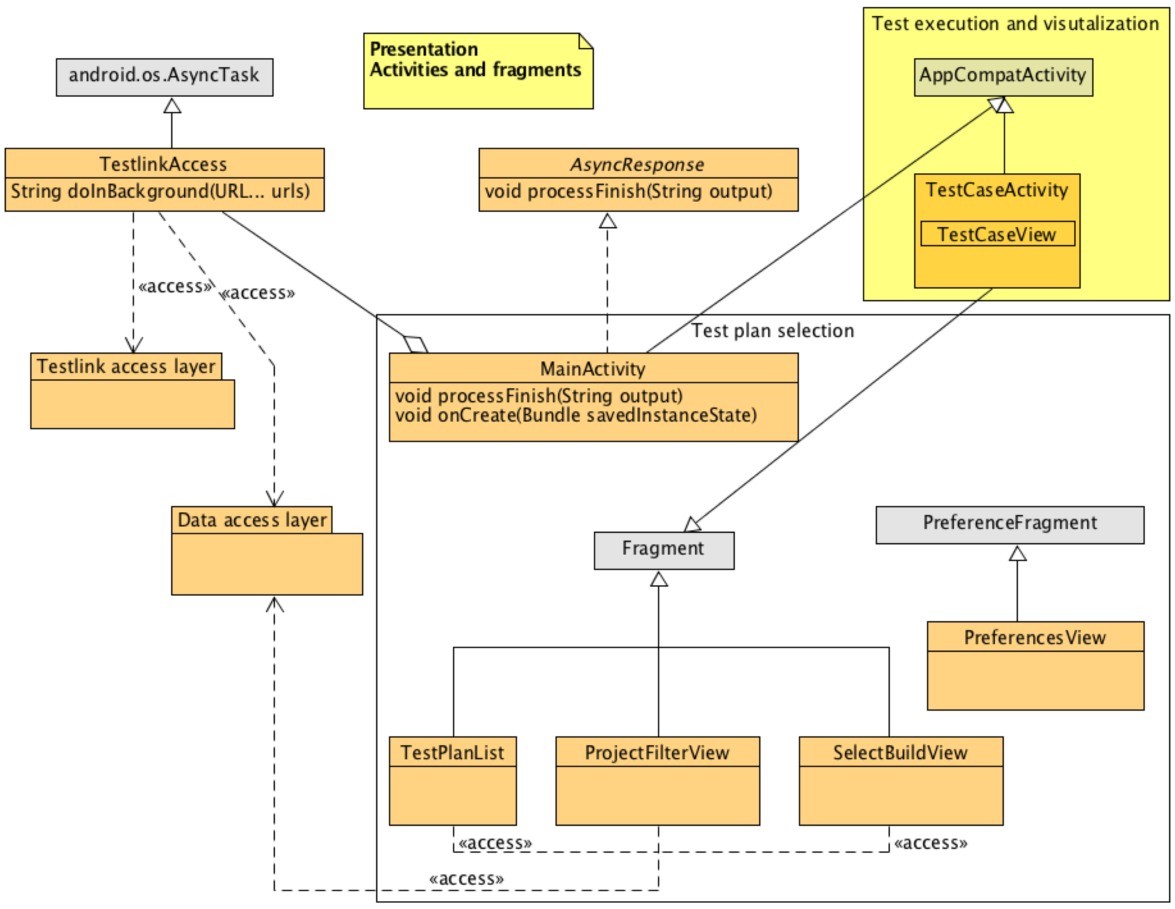
- Its useful in order to organize the testcase collection, but not that needed for a testplan scope. I don´t discard to still include it though.

Testlink access is wrapped using this library:

https://github.com/kinow/testlink-java-api

- This library is not used as it is directly from github, but it has been slightly modified instead.
- The library relies on apache *xml-rpc* client.
  - *Testlink* server exposes an *xml-rpc* API.
  - To include *xml-rpc* support, the project depends on:
    - *org.apache.xmlrpc:xmlrpc-client* (that adds another dependency)
      - *org.apache.commons:commons-configuration2:2.1*
- The models provided by this library have been reused as models for the database access layer.

Class diagram:



Fc

Description:

- *Test plan selection* roughly corresponds to the following use cases:
  - Test plan filter by project.
  - Display test plan list classified by project.
  - Show test plan details.
  - Preferences UI to set testlink access details.

The implementations of the different views are based on *Fragments* that are swapped within the *main activity*.

In order to move from one to another, a sliding menu has been included following the example in:

http://www.androidhive.info/2013/11/android-sliding-menu-using-navigation-drawer/

Moving back to the class diagram:

- *Test execution and visualization*, corresponds to:
  - Navigation of the test plan for reading.
  - Navigation of the test plan for execution.

Each test case is a fragment, but are rendered in a dedicated *Activity* (different than main *Activity*)

- The test case fragments are generated dynamically from the information about the test plan contents read from the db.

Some other design decisions:

- Access to testlink is implemented in a separated AsyncTask thread.
- DB connections are obtained using a singleton instance.

Testing

Following the terminology used by the official Android website, I have discarded to provide *local unit tests*. Reasons for that are:

- It would be suitable for utility methods, but the project hasn't many.

- Also suitable for *xml-rpc* testing of the testlink API, together with *mockito*, but since it has been taken from other project, its assumed that it will work flawlessly.
  - This assumption might be a bit risky, but in any case this part is going to be covered with manual functional tests.

What I plan to cover with automated tests:

- Instrumented Unit Tests to test the Dao persistence layer
  - The project already contains an example, *uoc.TestProjectDaoTest*
- Next, use these tests as utility methods to fill the db with data for automated integration tests:
  - Using Espresso

The *xml-rpc* interface with testlink it´s not going to be covered with any automated tests, it will be covered with manual functional tests.

All testing its going to be performed only with one device: a nexus 5x

- No effort its going to be made in order to make sure a wider device diversity

Additionally:

- The definition of the testplans and test descriptions are going to be created in testlink.
  - This will make easier the manual functional testing, taking advantage of the content in the teslink server.

Conclusion

I have to admit that my main motivation with this project was to recycle a bit my technical skills. I didn’t have any previous experience with Android development and I´m quite happy with what I have learned. Also I´m happy with the proposed project and with the achieved result. I wouldn't say that the outcome is a software that its ready for production, but in my opinion there could be a real practical use in this product.

Regarding what it has been the engineering process, I guess that the original estimations were quiet accurate, and it has been interesting to go over the different cycles of the project. I was already familiarized with some of the concepts from a professional environment, and there is some gap between the academic and the professional environments, but it has been interesting in any case. One issue that I had to face was the lack of appropiate tools: I had no Microsoft office, I used libre office as text editor, that wasn't a big obstacle, but I had to find tools for example to generate the GANTT diagram or to generate the Android Mocks. Depending on your position within the project, you usually have a set of tools that you trust and that you feel confident with them.

It took me some effort the kickoff of the project. I wanted to make sure the feasibility of the project, therefore I had to tune my original planning. Due my lack of expertise with Android, I ran some risk and in order to mitigate that, I tried to complete a prototype that was able to make a connection from Android to the testlink server right from the beginning. Additionally I wanted to have some prototype writing to the database in the early stages too. My original plan was to include the database persistence at the end, but I decided to amend it and have something working much earlier. In general most of the difficulties came from the lack of Android experience of the only resource (me).

It would have been great if I had had more time to add more testings, since I liked espresso a lot, and also to add more functionality, since from my point of view, the product could have some potential, but it needs more features.

Glossary

Espresso = automated UI testing framework

Junit = JUnit is a unit testing framework for the Java programming language.

SQLite.= SQLite is an embedded SQL database engine.

Testlink = TestLink is a web-based test management system that facilitates software quality assurance.

XML-RPC = XML-RPC is a remote procedure call (RPC) protocol which uses XML to encode its calls and HTTP as a transport mechanism

Bibliography

Due the fact that I didn't have previous experience with Android development, I have read a book:

Android 6 for Programmers:An App-Driven Approach (Deitel Developer)

- ***Editor:*** FINANCIAL TIMES/PRENTICE HALL;
- ISBN-10: 0134289366
- ISBN-13: 978-0134289366

I cannot say it was important for this project, neither I can recommend it, it was useful for me mainly to give me an insight.

What really has been useful for me it was using internet resources, the sources that helped me the most were the official Android development information center:

https://developer.android.com/studio/intro/index.html

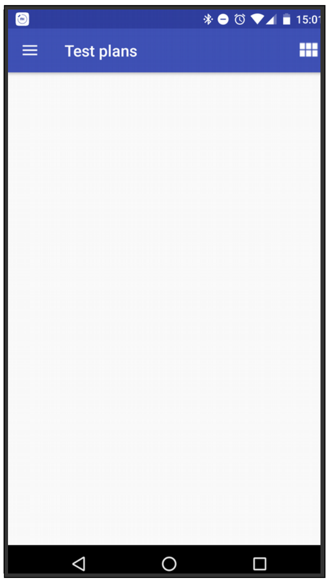
and, of course, stack overflow:

http://ww.stackoverflow.com

Appendix

User Manual

When the application is started, this is the first view that we have right after launch:

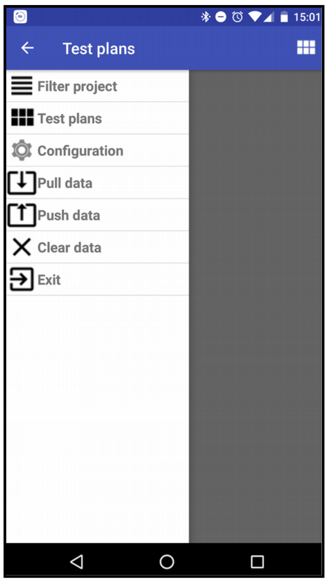


It correspond to the *testplan list view*, however the view is empty since it doesn't have any loaded data, we have to perform a user content synchronization first.

For this, we have first to go to the connection preferences. We reach that section using the *slide menu*

The ***slide menu*** is used to move across the different views of the application

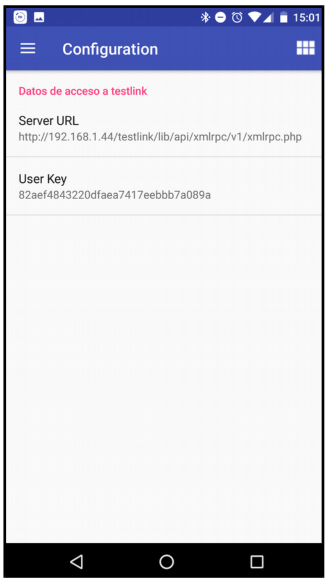
This is a view of the ***slide menu*** unfolded:



The slide menu is unfolded by clicking on the burger icon on the upper left corner. ☰

Once unfolded, we click on ***Configuration*** option ⚙

The ***Configuration*** panel:



Datos de acceso a testlink

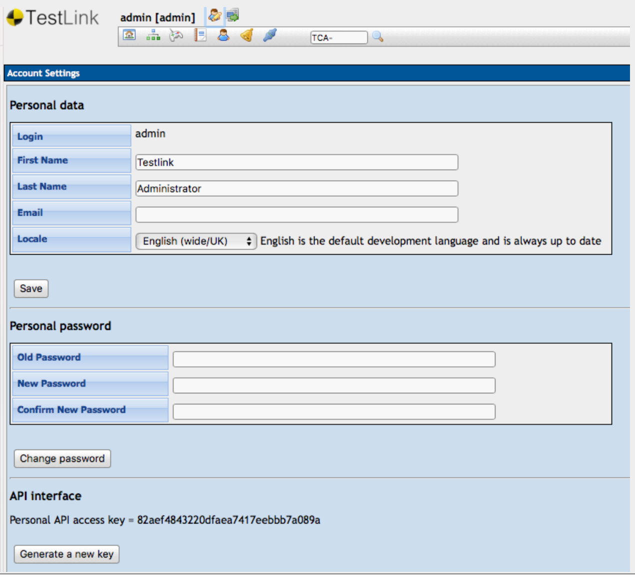
Server URL
http://192.168.1.44/testlink/lib/api/xmlrpc/v1/xmlrpc.php

User Key
82aef4843220dfaea7417eebbb7a089a

We have to make sure that we set the right **server URL and user key**

We can check the *user key* in testlink, from user settings option in the UI

See capture below:



Account Settings

Personal data

Login	admin
First Name	Testlink
Last Name	Administrator
Email	
Locale	English (wide/UK) English is the default development language and is always up to date

Personal password

Old Password	
New Password	
Confirm New Password	

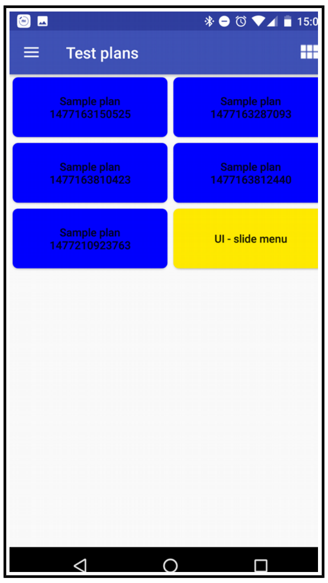
API interface

Personal API access key = 82aef4843220dfaea7417eebbb7a089a

Once we have made sure that we have the right connection settings, we can click on ***Pull data*** (on *slide menu)*

This will download the users test plans and save them locally

If we move back to the test plan view again, the list of test plans (on *slide menu)* is rendered:


15:0
Test plans
Sample plan 1477163150525
Sample plan 1477163287093
Sample plan 1477163810423
Sample plan 1477163812440
Sample plan 1477210923763
UI - slide menu

- Test plans are represented by buttons in this view.
- Note the different colors of the test plans, this will be clarified later on.

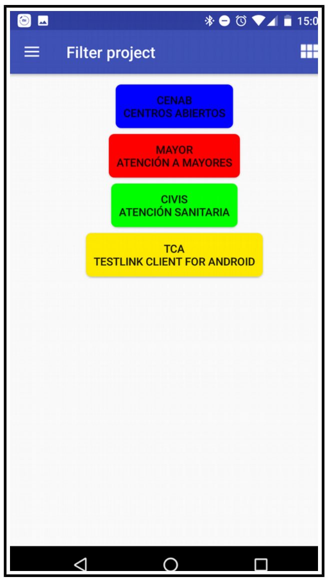
If the amount of test plans displayed is too large, we can limit it by moving to the *project filter view.* (on *slide menu)*

The project filter view shows a list of projects, where **each project can be either enabled or disabled.**

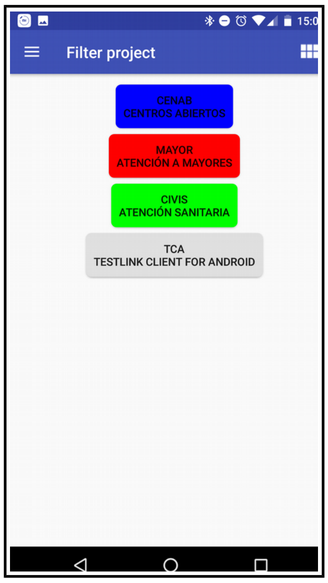
The list of test plans will display only the test plans corresponding to the projects that are enabled

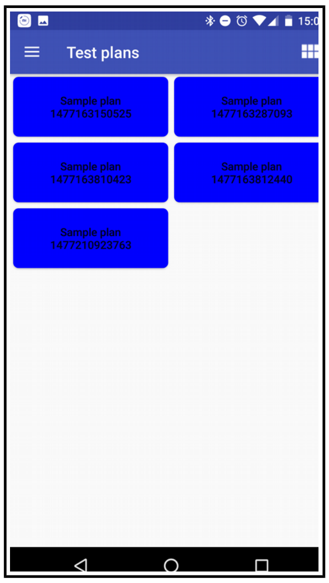
- Test plans are owned by projects in testlink

This is what the project filter view looks like:



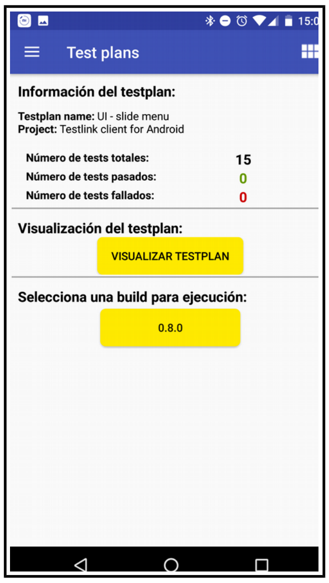
Lets see what happens if the yellow project is disabled and we move back to test plans view:





The yellow test plan has gone.

Next, if we enable the yellow project back, we open the test plan view we can click on the yellow test plan, and then test plan details for that test plan are displayed:



Test plans

Información del testplan:

Testplan name: UI - slide menu
Project: Testlink client for Android

Número de tests totales:	15
Número de tests pasados:	0
Número de tests fallados:	0

Visualización del testplan:

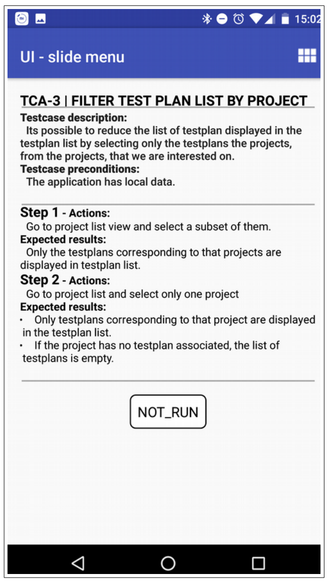
Selecciona una build para ejecución:

The details contain counters for the total amount of test cases that test plan contains and the number of passed and failed tests.

With the same color of the project, we see two buttons:

- **Test plan visualization** allows to go over the test cases on read mode.
- If we select a **build** instead, we get into test plan execution instead.

Lets try visualization first:



15:02

UI - slide menu

TCA-3 | FILTER TEST PLAN LIST BY PROJECT

Testcase description:
Its possible to reduce the list of testplan displayed in the testplan list by selecting only the testplans the projects, from the projects, that we are interested on.
Testcase preconditions:
The application has local data.

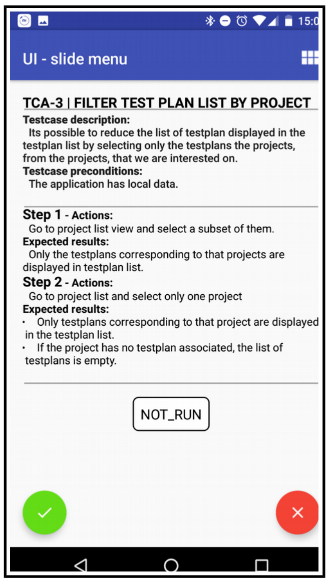
Step 1 - **Actions:**
Go to project list view and select a subset of them.
Expected results:
Only the testplans corresponding to that projects are displayed in testplan list.
Step 2 - **Actions:**
Go to project list and select only one project
Expected results:
- Only testplans corresponding to that project are displayed in the testplan list.
- If the project has no testplan associated, the list of testplans is empty.

NOT_RUN

We see the test case ID, name, description and test case steps.

- On the bottom, it can read what was the last execution result (or that it was never executed as in the capture)
- We can move from one test case to another swiping left or right.

Next, lets move back to test plan details (either by pressing the Android back icon or from the ▦ item **on the upper right corner**) and lets select a build instead, so that we get into test plan *execution mode*:



15:0

UI - slide menu

TCA-3 | FILTER TEST PLAN LIST BY PROJECT

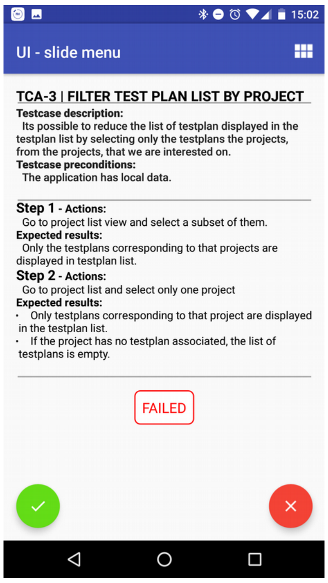
Testcase description:
Its possible to reduce the list of testplan displayed in the testplan list by selecting only the testplans the projects, from the projects, that we are interested on.
Testcase preconditions:
The application has local data.

Step 1 - **Actions:**
Go to project list view and select a subset of them.
Expected results:
Only the testplans corresponding to that projects are displayed in testplan list.
Step 2 - **Actions:**
Go to project list and select only one project
Expected results:
- Only testplans corresponding to that project are displayed in the testplan list.
- If the project has no testplan associated, the list of testplans is empty.

NOT_RUN

This time we see the same info as on visualization mode, but this time we have a **pass and a fail buttons on the bottom.**

If we set the test case to *failed*, the test case status changes into *failed state*:



15:02

UI - slide menu

TCA-3 | FILTER TEST PLAN LIST BY PROJECT

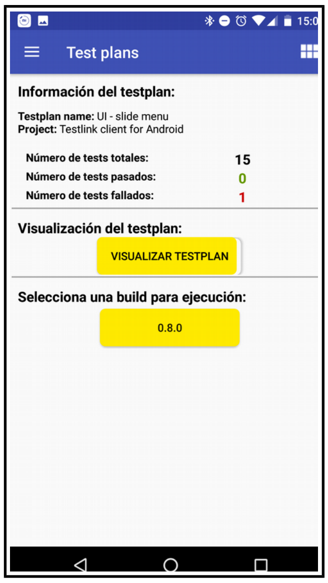
Testcase description:
Its possible to reduce the list of testplan displayed in the testplan list by selecting only the testplans the projects, from the projects, that we are interested on.
Testcase preconditions:
The application has local data.

Step 1 - **Actions:**
Go to project list view and select a subset of them.
Expected results:
Only the testplans corresponding to that projects are displayed in testplan list.
Step 2 - **Actions:**
Go to project list and select only one project
Expected results:
- Only testplans corresponding to that project are displayed in the testplan list.
- If the project has no testplan associated, the list of testplans is empty.

FAILED

Note that we can switch the state to pass. We can always change the state, but we cannot set it back to *NOT RUN* state.

Finally, if we move back to test plan details:



Test plans

Información del testplan:

Testplan name: UI - slide menu
Project: Testlink client for Android

Número de tests totales:	15
Número de tests pasados:	0
Número de tests fallados:	1

Visualización del testplan:

Selecciona una build para ejecución:

It can be seen that the counters of failed/passed test increase.

The results have been saved locally, in the Android device. Results are persisted among usages, and when we want to upload the results to the testlink server, we go to the slide menu and click on "*Push data*" (on *slide menu)*

Let´s remark again that its not the intent of the application to replace the corporative testlink server and centralize the test result repository, or use the application for test reporting, this is done using the legacy testlink server.

Installation Manual

The application is not available from the Google App store: its distributed as an apk and needs to be manually installed.

- This requires the target device to have the “Unknown sources” checked:

  Check the option Settings → Applications → Unknown sources.

Note that it has been generated using profile 25.0.0 and requires minimum 24 so Android device in which it has been installed needs to be *Nougat*

Note that development and testing has been performed in a *Nexus 5X*, its not assumed that it will work flawlesly in other devices.

Once the installation has been accomplished, please read the user manual to learn how to perform the synchronization with the testlink server.