Yes or No?

SWA Workshop 22 Project
Handout

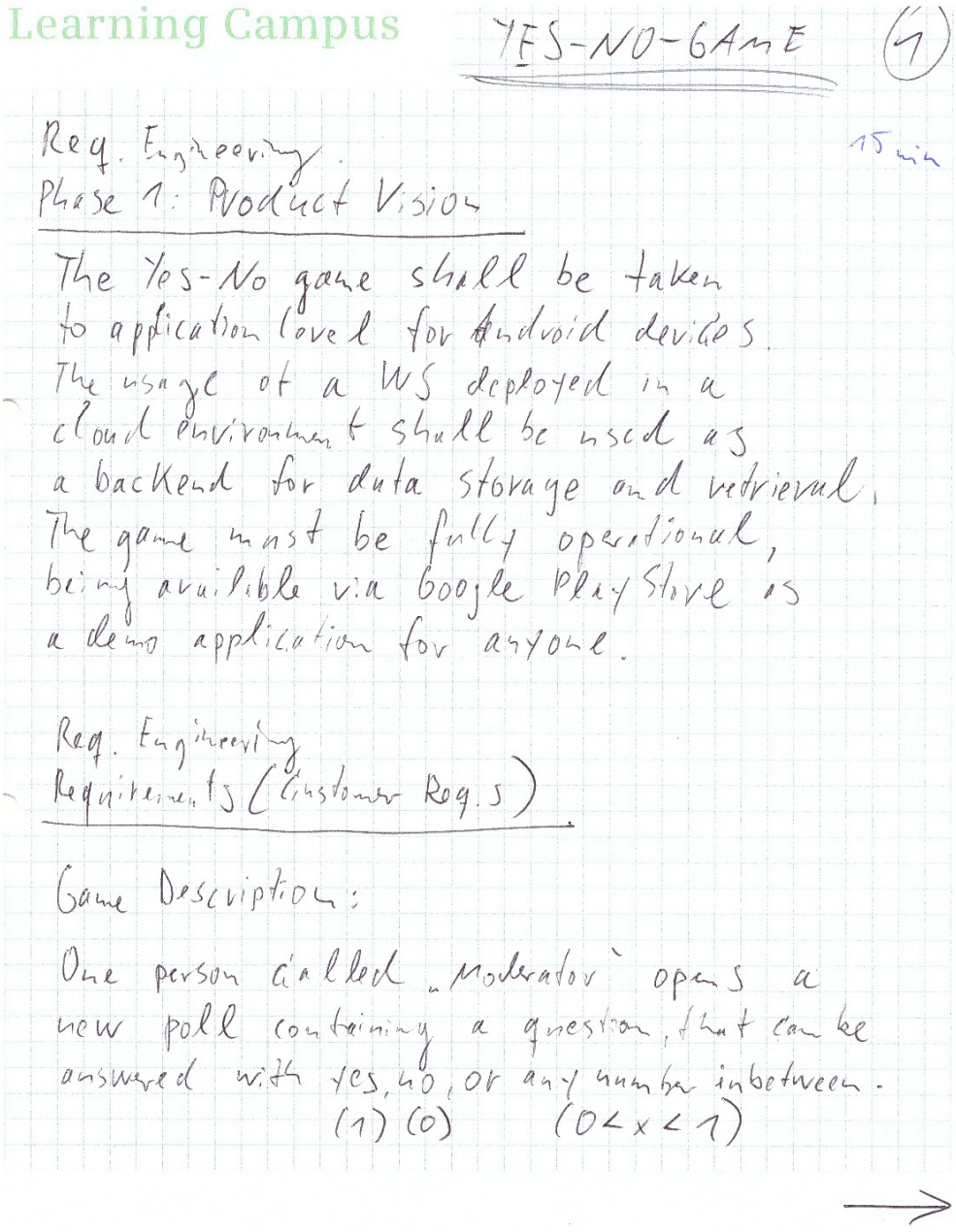
Hendrik.Stilke@siemens.com

Intro

In this document you will find some basic information about the project as well as some additional information on topics that were mentioned in the workshop curriculum. Things presented here are either to ease the understanding on what was said (mostly graphics) or they were too complex or off topic to put them in the small presentation.

I started sketching out the project on my way home from workshop 2 being in a train, then going on immeditaley in the two weeks after the workshop. This was real fun. Then the client was up and running with some mocking for the service. After the holiday season I continued and finished 1 week before the actual presentation.

The following 4 slides are my sketchout from the train. I did not bring everything into a more readable form, first to show you the original ideas, second because I do not see an added value in doing so.



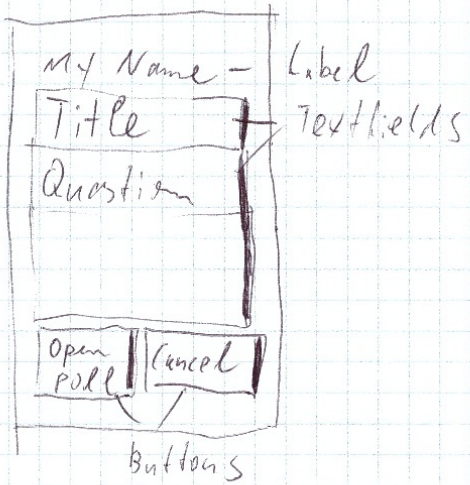
YES-NO-GAME (2)

20 min

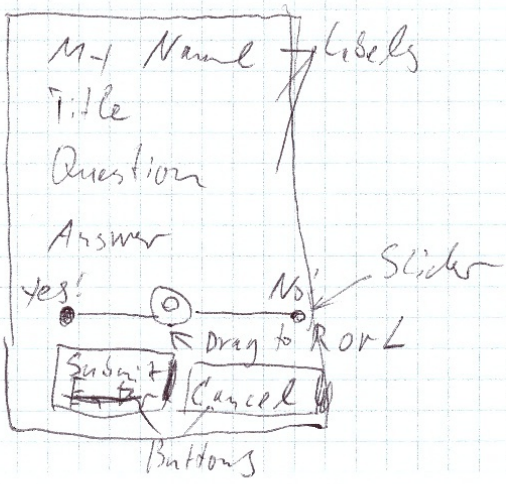
Other users can join the poll, by entering their name. They can answer the question by yes, no, or any value inbetween.
Any user may see the result of the poll after the „moderator" closed the poll.
The result can be shown as a list or by graphics.
The result is an ordered list of the participants ~~values~~. poll values.

Requirements Engineering
Application Sketch on f

Poll definition



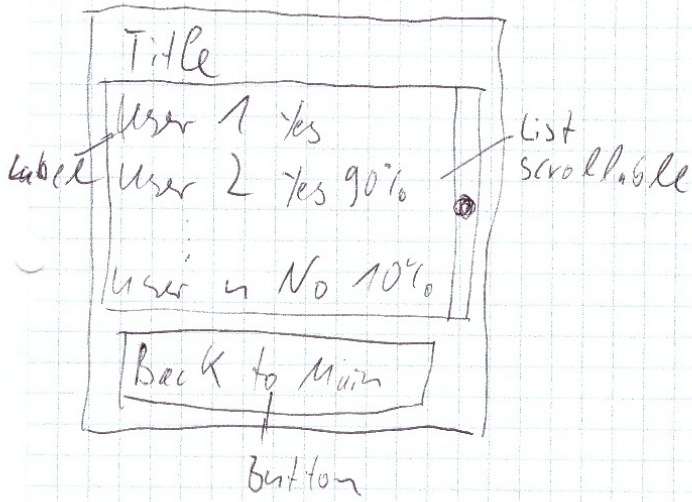
Poll user answer



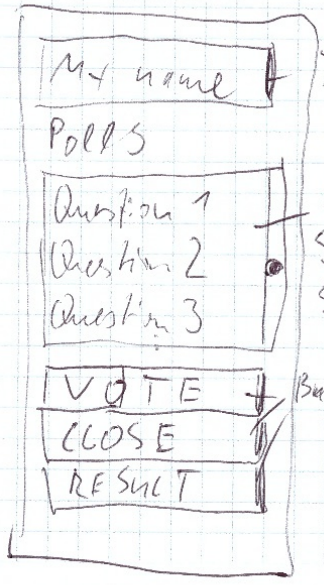
YES-NO-GAME (3)

10 min

Poll result screen



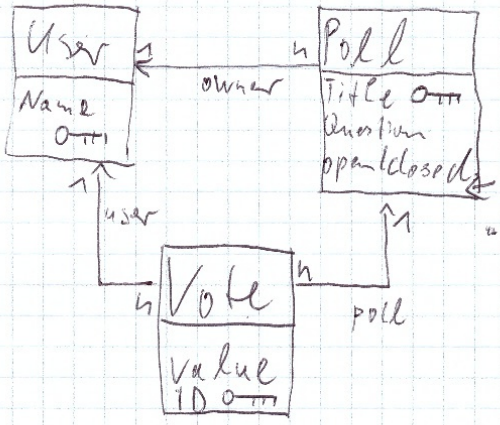
Main screen



YES-NO-GAME (4)

15 min

Design Phase - Domain Model:



Visualization using Lucid Charts: 1:15 h

Effort Estimation (Initial)

Conf. Requirements 7,5 h
- Description of Game 0,5 h
- Appl. Sketching 0,5 h
- Use-Cases, verbal 0,5 h

Design 2,5 h
- Domain Model 0,5 h
- Appl. Use-Cases 2 h

Implementation + Test = 29 h

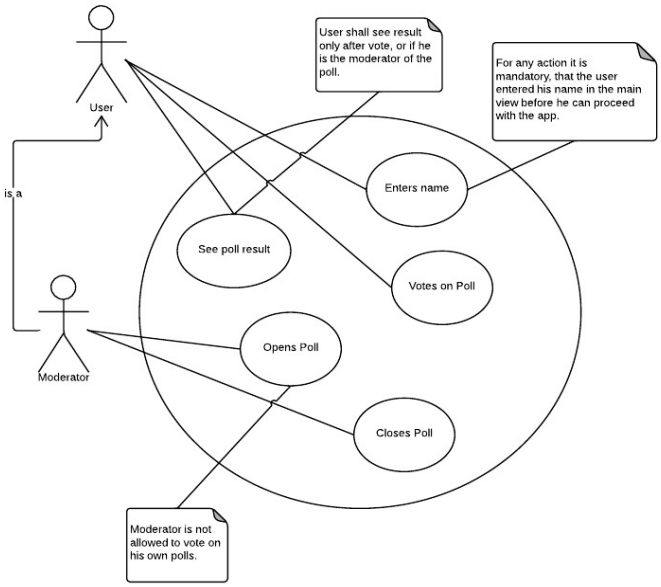
UI = 12 h
- Layout 4 h
- Business-Logic 8 h

WS = 5 h
- DB 1 h
- Domain 2 h
- Service 2 h

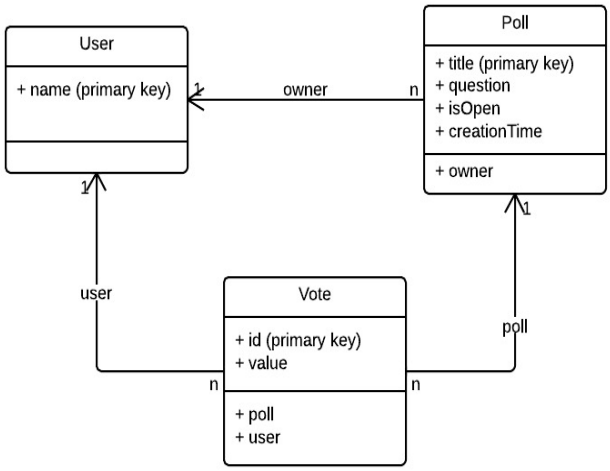
Connect UI to WS = 7 h
Integration 8 h
Test 4 h

At home I found it useful to do an effort estimation and to see how good or bad my skills in effort estimation are. Below in the document you will find a table with the results of this process. The sanity check form indicated that my estimation is worthless (5 points). So the result is mathematically speaking randomly good. Then I used lucidcharts (my favourite UML tooling for quick and dirty design) to get the ideas to a state where I could start coding the thing. Starting with

Use-case diagram:

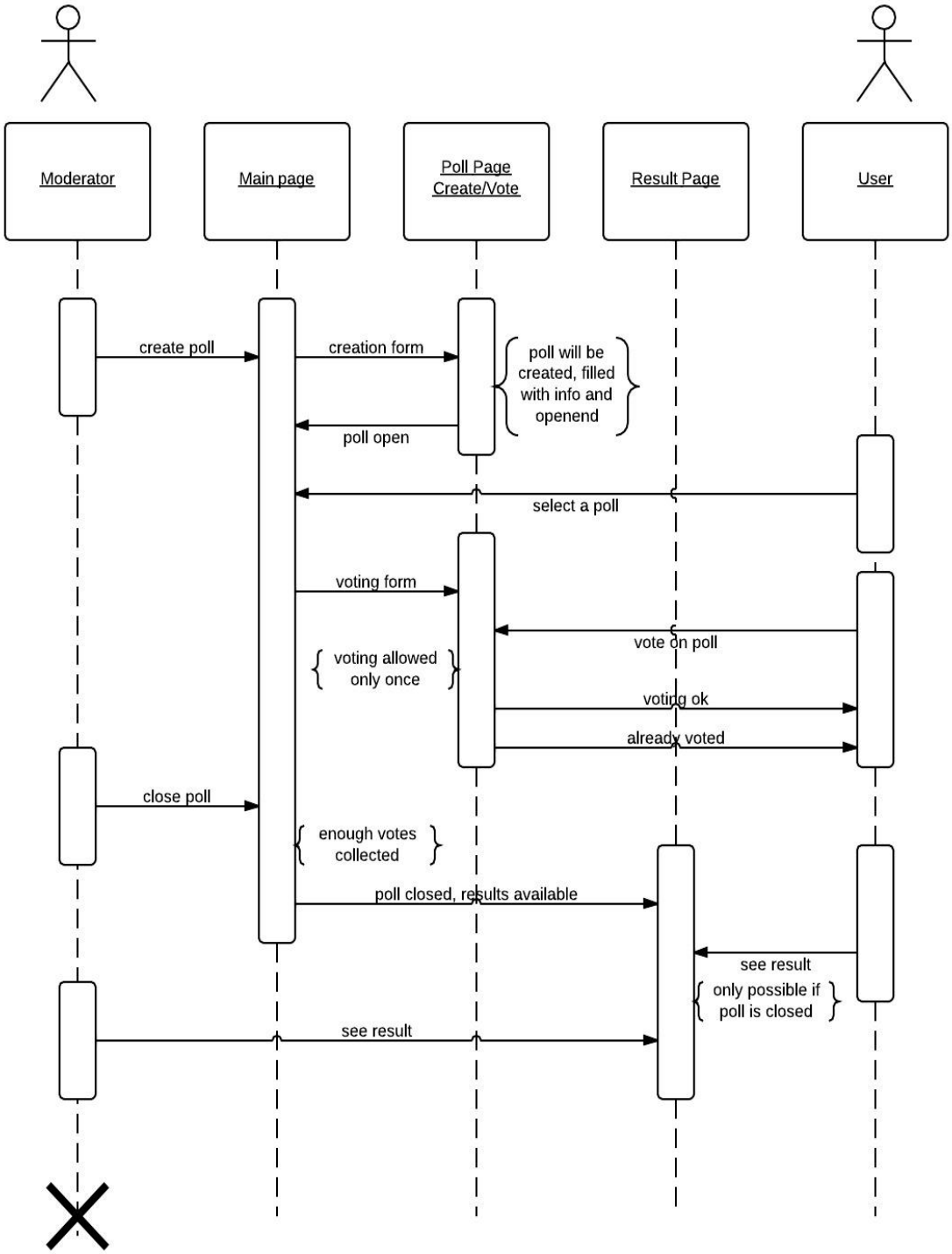


Data model (simple, isnt'it?)

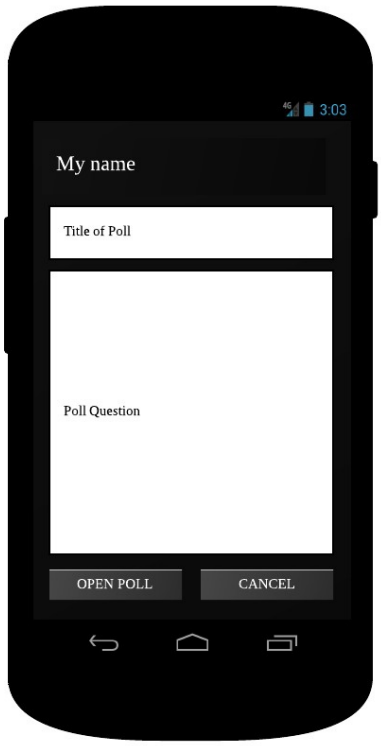


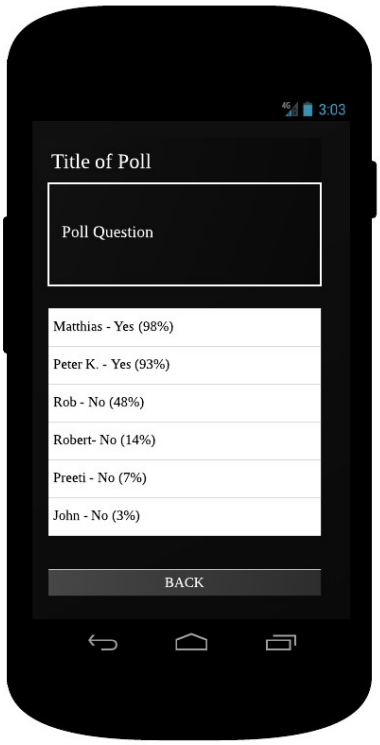
Sequence diagram for the lifecycle of a poll.

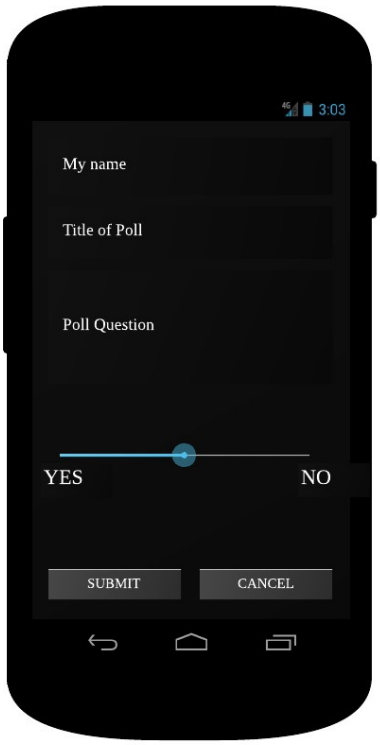
Remark: I did this, because if you do not fully understand what you put into a graphical form here, then you lack requirements or some other vital information is missing. It's agood chance you find problems in a very early stage of the project ;-)



Last but not least: **Activity diagrams** on how the UI design could look like.


3:03
My name
Title of Poll
Poll Question
OPEN POLL
CANCEL


3:03
Title of Poll
Poll Question
Matthias - Yes (98%)
Peter K. - Yes (93%)
Rob - No (48%)
Robert- No (14%)
Preeti - No (7%)
John - No (3%)
BACK


3:03
My name
Title of Poll
Poll Question
YES
NO
SUBMIT
CANCEL

Effort estimation and actual effort spent

I tracked my efforts spent during the project. Here are the numbers:

Task	Subtask	estimated	actual	finished?
Cust. Requirements				y
	Description of game / Vision	0,5h	20 min	y
	Solution sketchout	0,5h	25 min	y
	Use-case descriptions	0,5h	15 min	y
		1,5h	1h	
Design				y
	Domain model	30 min	15 min	y
	Use case formalization	2h	1h	y
		2,5h	1,25h	
Implementation				
	UI			y
	Layout UI	4h	6h	y
	Business Logic UI (incl Services)	8h	11h	y
		12h	17h	
	Webservice			y
	Set up DB	1h	30 min	y
	Implement domain classes & DTO	2h	3h	y
	Set up service (going online)	2h	2h	y
		5h	5,5h	
	Connect UI + Webservice			y
	Final Integration	8h	16h	y
	Testing	4h	2h	y
		12h	12h	
	TOTAL	33h	42,75	y

Lessons learned

Weak points (red) / wasted time

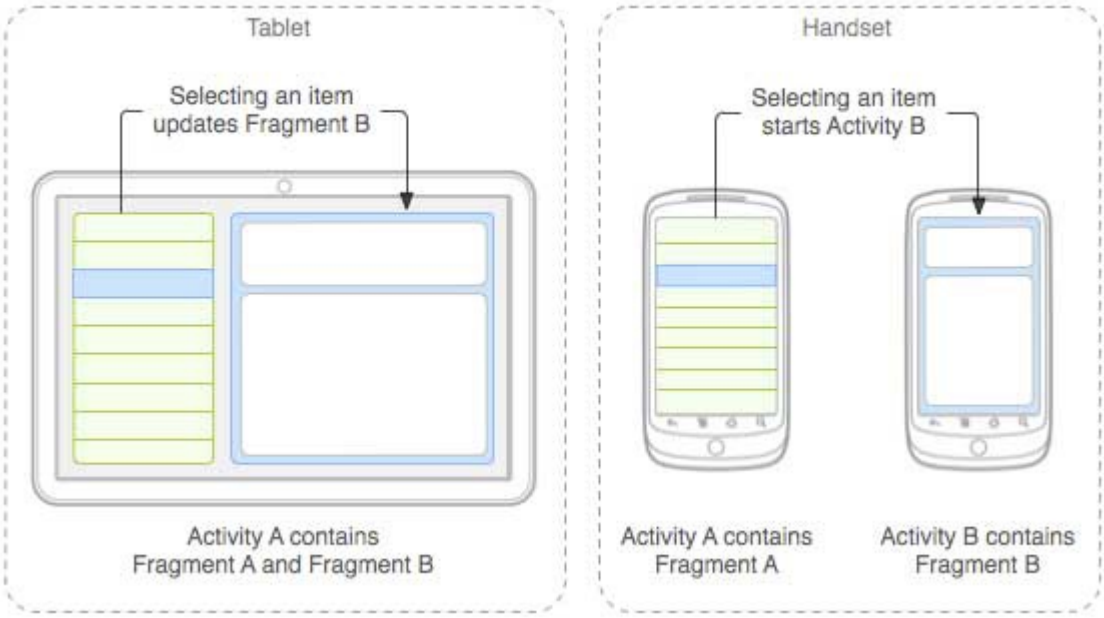
- Layout UI : Extra effort for colored background of poll list. Learning curve.
- Business Logic UI : I was slower than I estimated. Services needed extra implementation for "simple" types.
- Web service domain classes: Problem with PUT (not solved yet). Higher learning curve with "Flight PHP" framework.
- Final integration: some more bugs which had to be found by debugging sessions
- Adapting to Rob's service implementation took additional 6h (Thats what happens if you change your interfaces last minute!)

Strong points (green) / gained time

- Requirements were pretty clear right from the start
- Design: Tooling was effective (lucidcharts.com used)
- Set up DB: Took less time than estimated. No 2nd run needed.
- Final testing: Only one bug found (OK, you will find some more)

Why using multiple activities in one application?

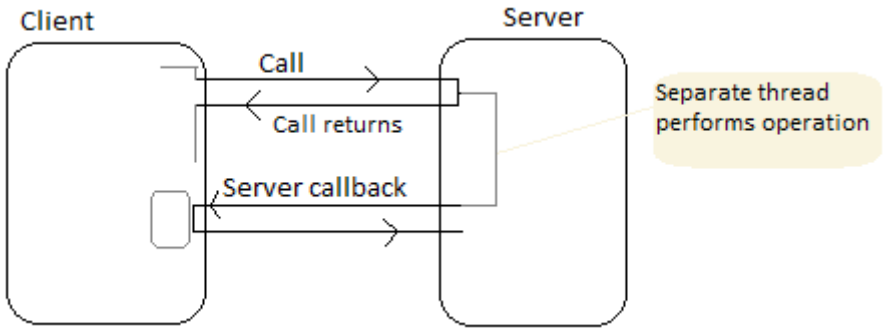
One reason is that you can build applications for tablets that would not work if you change the UI by hand loading all the components (e.g. using an xml layout reader). If you use activities and fragments, life gets easier here.



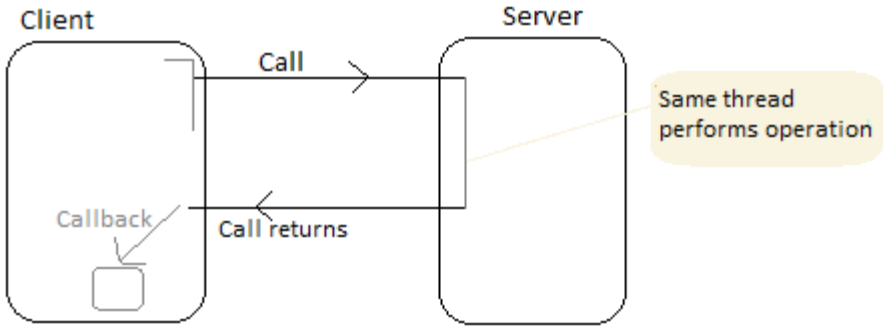
Calling a remote service asynchronously

Just in case you are not familiar with that pattern.

This is how it looks like from the client side:
The task object starts a separate background thread to the server, preserving a callback listener for the result. The new thread asks for an operation on the server side (request to the REST service in our case). When finished the server responses the result.
The background thread then calls the listener delivering the result.



The server side is easy and straightforward:



Activity lifecycle diagram