System Design Document

DAT255, Software engineering project
Chalmers, VT 2012

Patrik Ackerfors	861017-4990
Christoffer Kjernald	870820-5516
Erik Olesund	880917-2490
Markus Ulenius	880320-5635

1 Introduction

Detta dokument syftar till att introducera medlemmar av projektet på ett snabbt och överskådligt sätt. Dokumentet ger stöd till utvecklarna under implementationerna genom att beskriva designen och förhållandet mellan dess beståndsdelar, hur de förhåller sig och kommunicerar med varandra. Klasserna är namngivan och beskrivna så att utvecklare ska få en god överblick över appens struktur.

1.1 Design goal

Kärnan i applikationen rör smart och innovativ hantering av listor. Med hjälp av denna app finns möjlighet att skapa en lista och sedan dela med sig av den med andra användare av applikationen. Användarna som får listan delad till sig kan välja att ”boka” en artikel på listan. På så sätt får de inbjudna användarna möjlighet “boka” varje artikel så att hela listan är täckt. Skaparen av listan kan välja huruvida denne ska kunna se vem och vilka artiklar som blivit “bokade”. Tänkta användningsområden är till exempel brölopp och födelsedagsfester. Också mer vardagliga event såsom en grillfest kan dra nytta av denna app genom att enkelt skapa ett knytkalas där alla gäster bidrar till kvällens inköpslista genom att “boka” och inhandla ingredienser. Med andra ord är denna app främst tänkt att lösa problem som rör gemensamma event där artiklar/produkter ska köpas och där det är önskvärt att likadana produkter inte köps flera gånger.

Målet med designen som släpps inom ramen för denna kurs är att designen ska vara av så skalbar karaktär som möjligt. Det ska vara enkelt att bygga ut den med ny funktionalitet så att den blir mer användarvänlig ach användbar. Designen ska vidare möjliggöra enkla tester så att buggar enkelt kan spåras och åtgärdas. Ett annat mål är kodningen ska kunna genomföras så effektivt som möjligt med utgångspunkt i designen.

1.2 Definitions, acronyms and abbreviations

- **Lists** är benämnningen på listorna som skapas i applikationen. Användarna skapar en eller flera listor.

- **Items** är benämningen på listornas innehåll, det vill säga de artiklar som finns i respektive lista.
- **Book** är benämningen på att boka en artikel och markera den som köpt eller införskaffad

1.3 References and 3PP

Referens till grunden för vår mailkommunikation (klassen Mail.java): Av Jon Simon finns på följande internetadress:
http://www.jondev.net/articles/Sending_Emails_without_User_Intervention_(no_Intents)_in_Android
Detta är under 3PP-copyright. Vi har också använt två javabibliotek: activation.jar samt additionnal.jar.

2 Proposed system architecture

2.1 Overview

Applikationen innehåller sex olika aktiviteter vars vyer reagerar på knapptryck och pek på skärmen. Dessa aktiviteter är utförligt beskrivna i pdf-filen *Activities.*

De tre aktiviterna som visar "data", det vill säga MainListActivity, MainItemActivity samt ItemDetailsActivity kommunicerar med en databasadapter med namn ListsDbAdapter och ItemsDbAdapter beroende på om listor eller artiklar i listor ska visas. Dessa adaptrar har i sin tur har metoder som gör anrop mot en lokal SQLite databas och returnerar data från databasen. Editor-aktiviteterna använder endast data från den aktivitet som startade den.

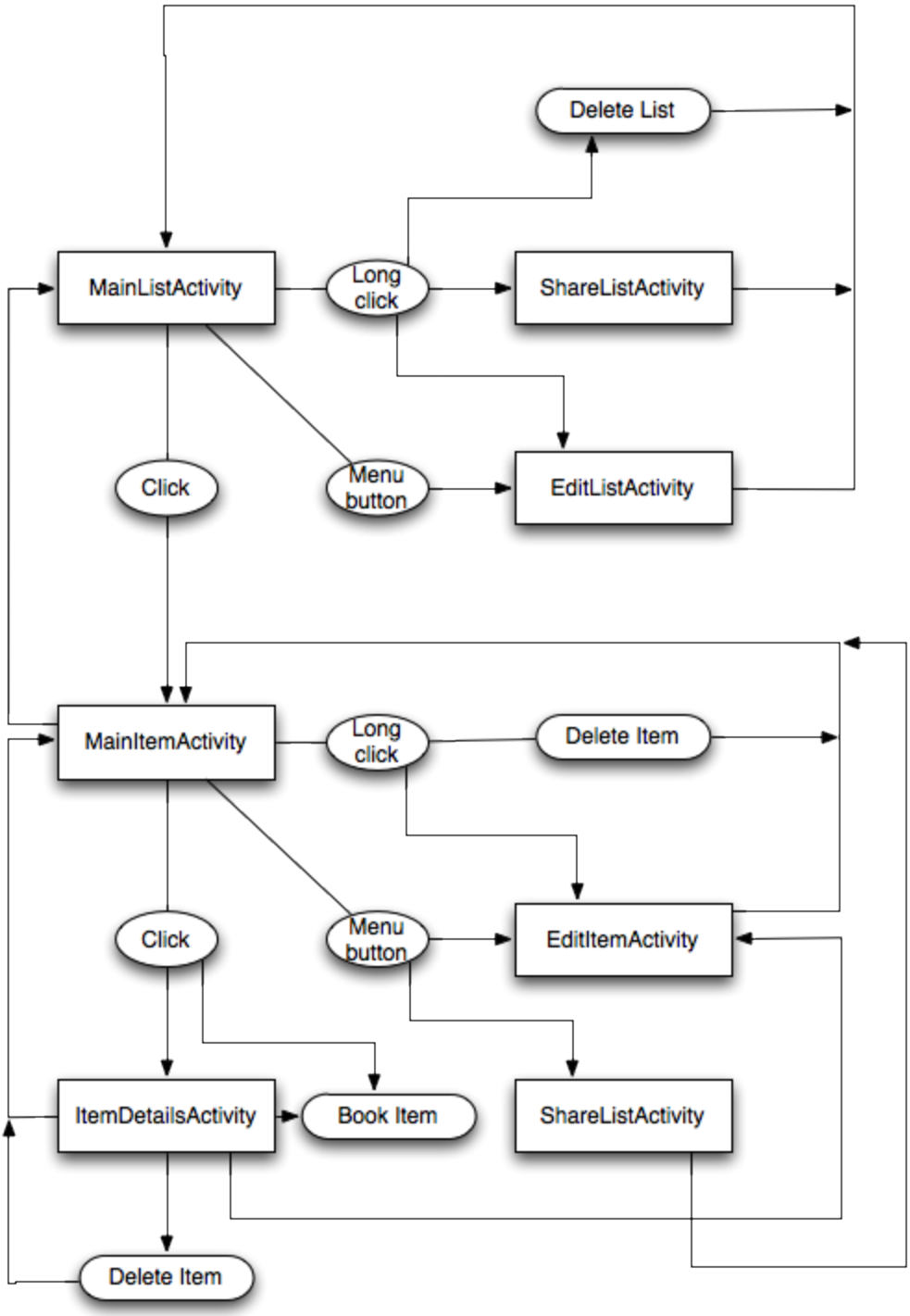
2.2 Software decomposition

2.2.1 General

Applikationen är uppbyggd, som tidigare nämnt, genom sex olika aktiviteter:

- MainListActivity
- MainItemActivity
- ItemsDetailActivity
- ListEditor
- ShareListActivity
- ItemEditor

Nedan visas ett flödesdiagram hur en användare av applikationen kan navigera mellan dessa.



2.2.3 Communication

Inom applikationen sker kommunikationen genom att intents skapas och den data som nästa aktivitet behöver skickas via bundles som “extras” de vill säga den extra information som man själv har valt ska skickas med.

Kommunikation utom applikationen finns möjlig genom att inifrån applikationen skicka ett e-mail som innehåller en lista. På så sätt kan andra personer, som inte har tillgång till applikationen tillgång till informationen. Det räcker med att de har en fungerande e-mailadress. Detta görs med hjälp av en SMTP-koppling och ett Gmail-konto och hanteras av klassen Mail.java (copyright Jon Simon, 2010) samt ShareListActivity.java.

2.2.4 Decomposition into subsystems

N/A

2.2.5 Layering

N/A

2.2.6 Dependency analysis

Aktiviteterna är löst koppla till varandra och kan ses som modulära. De fyra aktiviteter som skickar frågor till databasen, MainListActivity, MainItemActivity, ItemDetailsActivity och ShareListActivity kan ses som enskilda applikationer, medan de sista två förlitar sig enbart på den data som skickas till den via Intents.

2.3 Concurrency issues

Detta eventuella problem är egentligen ej applicerbart för vår applikation. Den enda nätverkskommunikation som sker är att ett mail skickas från ShareListActivity.java, och appen bekräftar därefter om mailet kunde skickas eller ej.

2.4 Persistent data management

Applikationen sparar data lokalt i SQLite. Vid första körning av applikationen skapas två tabeller, lists och items. Lists sparar data relatierat till de listor som användarna skapar och items sparar data om de items (artiklar) som användarna sparar.

Lists ser ut som följer:

_id (INTEGER PRIMARY KEY AUTOINCREMENT)	**listTitle** (TEXT NOT NULL)
1	Patriks födelsedag
2	Sannas bröllop
3	Storhandling

Items ser ut som följer:

_id (INTEGER PRIMARY	**itemTitle** (TEXT NOT	**description** (TEXT)	**booked** (INTEGER)	**parent** (INTEGER)

KEY AUTOINCREMENT)	NULL)			
1	En japansk kniv	Helst av märket Kai Shun	0	2
2	En androidmobil		1	1
3	Rödbetor	Kravmärkta och små	0	3
4	Bestick	Helst 12st av varje.	0	2

När användaren klickar på en lista skickas dess _id med genom Intent. I nästa aktivitet, MainItemActivity, skickas en databasförfrågan med detta _id i som ett WHERE parent = _id (från lists). När användaren skapar en ny item får denna item samma parent-nummer som listans _id. På så sätt går det alltid att knyta en item till en lista.

För varje item sparas information om denna är bokad eller ej där 0 är inte bokad och 1 är bokad.

2.5 Access control and security

N/A

2.6 Boundary conditions

Systemet startas bara genom att appen öppnas, och stängs när man lämnar den. Varje aktivitet ser till att inga öppna databaskopplingar lämnas när aktiviteten lämnas. Misshandel av mjukvaran kan självklart leda till fel och kraschar, även Gmail-kontot måste fungera för att applikationen ska fungera utan problem.