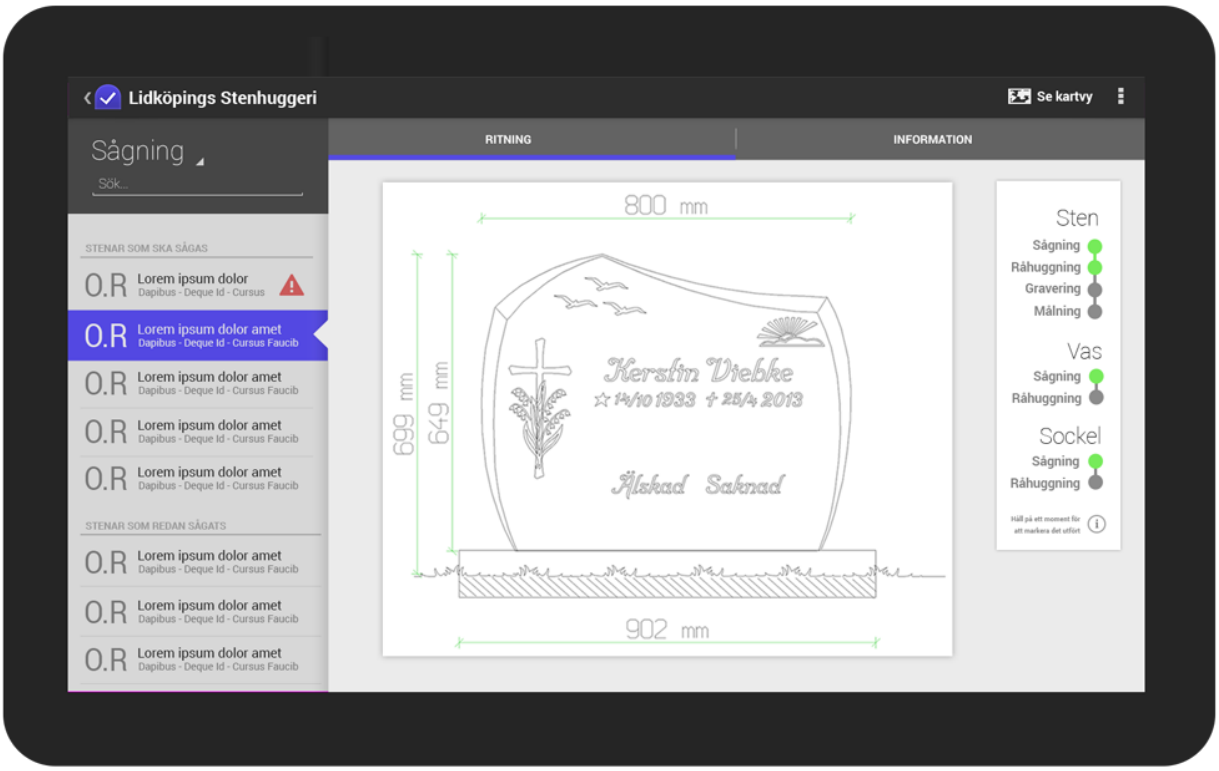
Developer Manual

Lidköpings Stenhuggeri, 2013-10-22, v2

Lidkoping-SH applikation

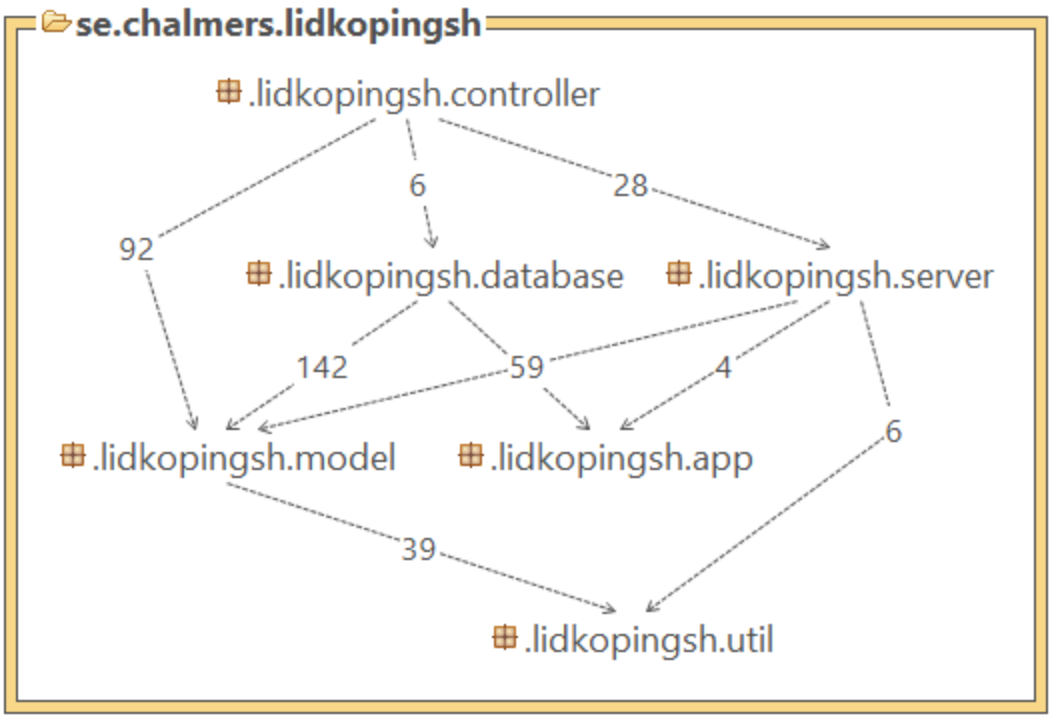


Introduction

This document contains helpful information for developers of the application.

Table of Contents

Overview


se.chalmers.lidkopingsh
.lidkopingsh.controller
6
28
92
.lidkopingsh.database
.lidkopingsh.server
142
59
4
6
.lidkopingsh.model
.lidkopingsh.app
39
.lidkopingsh.util

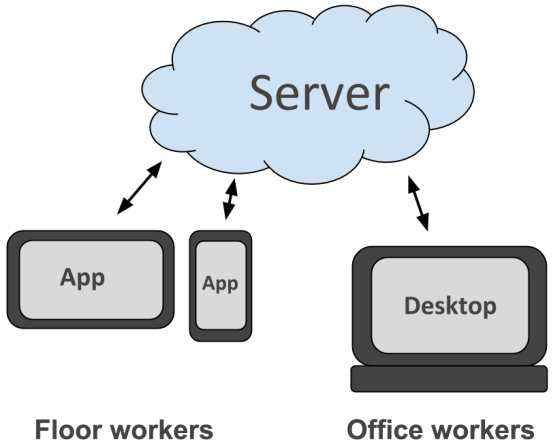
Research and analysis

Field research

Visited the working site of Lidköping Stenhuggeri in order to make sure that the application fit the customer's desires.

Vision

See picture for the app environment.



Target unit

Lidköpings Stenhuggeri will buy the Acer Iconia A1-810 tablet for use within the company. It is a tablet with an eight inches screen, one GB of ram, a quad processor clocked at 1,20 Ghz and a resolution of 1024x768.

The lowest supported version of Android supported by the app is 4.0 Ice Cream sandwich. However the app is partly designed to make it easy supporting older devices in the future by using the Android Support library instead of new Apis introduced in newer versions of android. The only thing that prevents older units as far back as Android 2.2 Eclair from running the applications right now is a GridLayout used when showing details about an order.

Targeting Android tablets running version 4.3 Jelly bean, with support for tablets running version 4.0, Icecream Sandwich and above. Originally aiming for 10,1" screens, now targeting a 7,9" screen as well as mobile devices. There is no reason for the application to target devices with older version than 4.0 since most of the devices on the market today uses 4.0 or newer and the users we are aiming at will be using compatible units.

Personas

For more information, see Personas documentation.

Build and install instructions

Requirements

Android SDK, including Android Development Toolkit (ADT) and Eclipse.
Everything you need to get started: http://developer.android.com/sdk/index.html

Google APIs

If you already have eclipse, use this link to download the Android Development Toolkit:
http://developer.android.com/tools/sdk/eclipse-adt.html

For more information about Android development, visit:
http://developer.android.com/tools/index.html

For better performance while running the Android emulator, if you are on an Intel processor system, install the Intel HAXM (Hardware Accelerated Execution Manager) extension through the Android SDK Manager.

A PHP web server for the API service and a MySQL database for the central data storage. The web service must be accessible (i.e. configuring IP-address/DNS, firewall settings) from the environment the Android application should be used.

Buildinstructions for the client

Clone the project on github and import it in eclipse. All external libraries are bundled with the project in the /libs/ folder. Two of them needs to be imported as library projects. Those are Google Play Services and Google Maps Utilities. To be able use the MapView you also need to use the custom keystore named debug.keystore in the repository's root folder which includes an accepted API key to Google Maps.

You must send a request for a membership in the Farenheight group on GitHub in order to be able to push changes to the project.

Github repository: https://github.com/Farenheight/Lidkoping-SH.git

Build instructions for the server

The web server REST API is running on a PHP web server. Publish the source files (folder api) in the root folder, exactly as they are on Git. A MySQL database should then be created and then populated by the SQL file in /api/setup/ folder.

The PHP application uses a config file, where the MySQL host address, username, password, and database name must be specified. That's also where some other secret settings is stored. This file is named db_config.php and is located in the root of /api/. In the Git repo is an example.

The folder /web_gui/ contains the web interface source code. This can be published to the same web site as the API, or another by using a separate db_config.php file to setup database configuration.

In index.php in api folder, where requiring a HTTPS connection for secure transportation. Modify it to suit your needs.

Installation

To run the client application install the distributed .apk file in the /dist/ folder in the repository. Every release on github has an attached .apk file as well.

Application Design

Technologies

- Android application written in Java together with Android for graphical user interface.
- Server application written in PHP with a MySQL database.
- SQLite database in Android.

Design Patterns

The application uses an independant model separated from the database, controllers and views. Views are static files in XML format, modified by controllers.

This design is somewhat closely related to the MVC (model-view-controller) pattern where the big difference is the android part. The controller and view-separation is in some ways diffuse.

Important Design Decisions

Interaction design and GUI

Recently plenty of well-known apps in the Android ecosystem such as Gmail, Evernote and the Android Contacts app all have turned to a two-pane layouts on tablets. Among other things to make better use of the bigger screens. Our app fits perfectly with this design pattern and it makes it easy and quick for users showing an order and then instantly switch to browsing the list to look for another one.

Data management

The application utilizes that all components of an android app runs in the same process and therefore all can access a global singleton. Especially handy in our case since we have complex and non-persistent objects with data that would be time consuming and hard to pass around different activities by serialization. Google mentions this as a prefered solution in their developer manual.[1]

Handler

Using a handler-class to create and administrate the model and database. Closely related to a factory.

Utilities

Using a helper for syncing different kinds of lists by comparing them and returning an updated combined version.

[1] http://developer.android.com/guide/faq/framework.html

Listeners

Using a listener-class with the method changed which is called from the controller every time an object is going to be updated in the model followed by an update of the database, locally and remotely.

Database Layer

Using a DatabaseLayer between the model and the local database in order to keep SQL queries outside of the model.

Web service

A web service with a central database storage to synchronize data between all endpoints. The web service is written in PHP, used as a REST API. Data is formatted with JSON. Requests and responses are UTF-8 encoded. Se attached Web Service API Documentation for more information about using the service.

Server Connector

Using a ServerConnector between the internal modules and the remote server in order to keep logic handling http queries and JSON objects outside of the core.

The class ServerConnector uses Android AsyncTasks to run network operations concurrently to the main thread. The methods are divided into different classes: AsyncTaskGet and AsyncTaskSend where the first is used for collecting updates from server and the second one for sending updates to the server.

Server - Client communication

While getting updates from server, we are sending an array, where each post contains another Long array containing two values:
1. Order’s Identification Number.
2. Order’s Timestamp, last time updated on client.

These values are used to identify whether an order is going to be updated or not. Identifying the order by id, and if to be updated or not by the timestamp.

The client handles conflicts locally by getting updates before applying the change and sending it back to the server, if first try to send change fails.

Model synchronisation

Every time the application gets data from the server, the model is synced with all data sent from the server. Everytime the application tries to change any order, the data is fetched from the server and then the changes are applied. This makes sure that no conflicts should come up. It also means that the model can only be changed when the device has internet access.

All model classes have synchronization methods with the class itself as a parameter. This method changes all of the object's content without replacing object references and without replacing listeners.

GIT - Branches

A new branch has been created for every new feature.
How to follow branching guidelines: http://nvie.com/posts/a-successful-git-branching-model/

- **Dev**

Main branch, to be merged into with every new feature branch when the feature implementation has been successfully completed.

- **Master**

Containing a new commit for each release, with a tag.

Addons

FindBugs
Finding potential bugs.

Structure Analysis (STAN)
Finding potential two-way-dependencies in Eclipse.

EGit
Administrating git in Eclipse.

Libraries

Google Gson

Parsing a JSON string into an Object of choice.

More info: https://code.google.com/p/google-gson/

PhotoView

To make it easy to the add zoom and pan features to the blueprint shown on each order the PhotoView library is used. We looked at a few different libraries, but PhotoView had two major adavantages. The first being that it can be attached to an existing `ImageView` object in android which makes it easy to replace the library later on if necessary. It also means that even if the library would fail to load of some reason the blueprint would still be shown. The second major advantage of PhotoView is that it currently is under heavy development and fixes and updates are released almost on a daily basis.

More info: https://github.com/chrisbanes/PhotoView

Google play-services

To be able to show where the company's orders or stones are geographically the Google Maps Android API v2 is used which requires Google Play Services. Lack of alternatives and that Google Play Services is bundle with the Android SDK made the choice fairly easy. However, we have had a really good experience with the Google Maps Api.

More info: http://developer.android.com/google/play-services/

Android Maps Utils

Google Maps doesn't support adding custom text labels on each marker. To make it easier for users of the application to see what orders were shown in the map view we wanted to add the order notation to each marker. Android Maps Utils provides methods for doing just that and as far as we know there is no good alternatives. The library is developed by Google by people also in the Google Maps Android team. We decided to use v 0.1 even though 0.2 was released recently. The reason is that the new version changed a lot and the documentation wasn't updated yet which made it hard making it work. We didn't need any of the new functionality added either making version 0.2 bigger than necessary.

More info: http://googlemaps.github.io/android-maps-utils/

Android remote stacktrace

To be able to see how stable the app is when deployed on non-developer phones, the Android remote stacktrace library is used. It automatically sends stacktraces of uncought exceptions to our server.

Development methodics

Agile Development

- SCRUM
- Xtreme programming