Project Plan

Introduction

About Buddycloud

Buddycloud is an Open-Source distributed online social network. Its protocols and specification allow anyone to host their own buddycloud und connect to other buddycloud servers and share content among their users.
The Buddycloud Community is working the Buddycloud server (implemented in various different languages), an HTTP API server, a native webclient and native mobile clients to allow everybody to setup their own hosting service very fast.

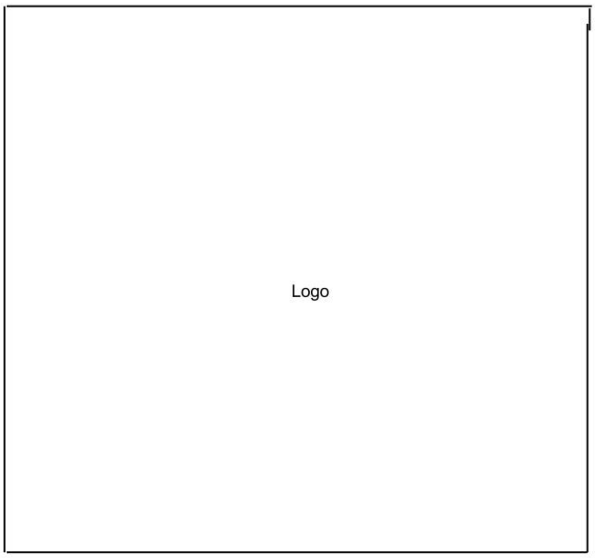
About our Project

In this project we will implement a basic prototype Android client-application for Buddycloud Http-API-Servers. It is part of the Internet of Services Bachelor Project at TU Berlin and will implemented by Marvin Byfield, Siyun Li and Yuwen Chen and supervised by prof. dr. Peter Ruppel ~~of the DAI-Labor~~.

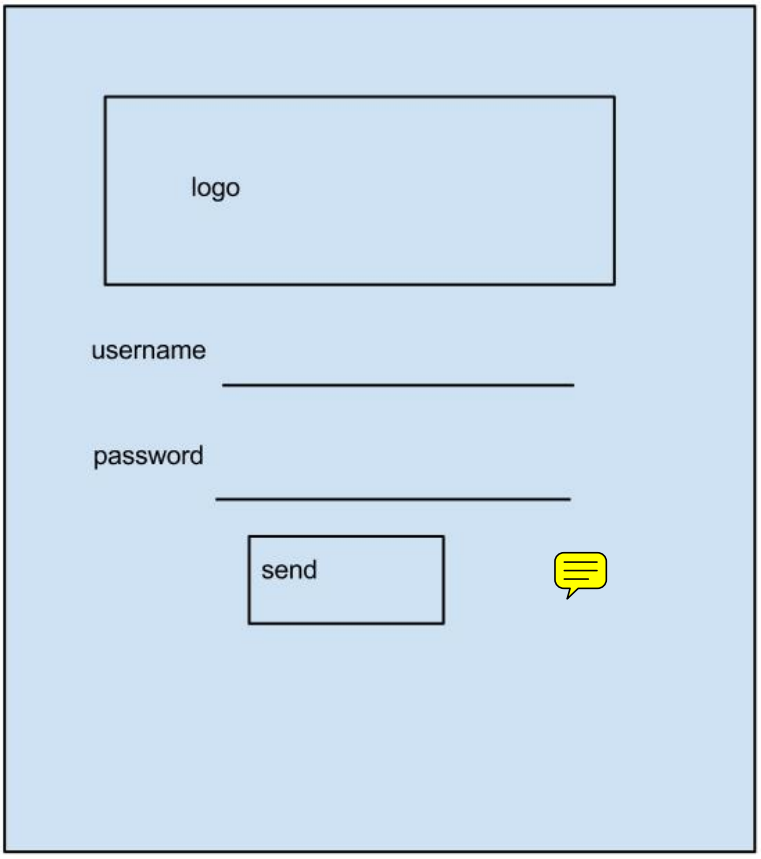
The basic version of the client will be able to connect to the users home buddycloud http-api-server retrieve various information and display them. The prototype will we used by the buddycloud community to evaluate the workload for implementing client-applications to the http-api compared to xmpp and might be improved to become the native android-client published by buddycloud.com in the android appstore and it will be used as proof of concept for any followong Android implementation based on the buddycloud http-api.

Features

For an overview of the basic features and architecture please have a look at the following wire sketches. Features in brackets are optional improvements.

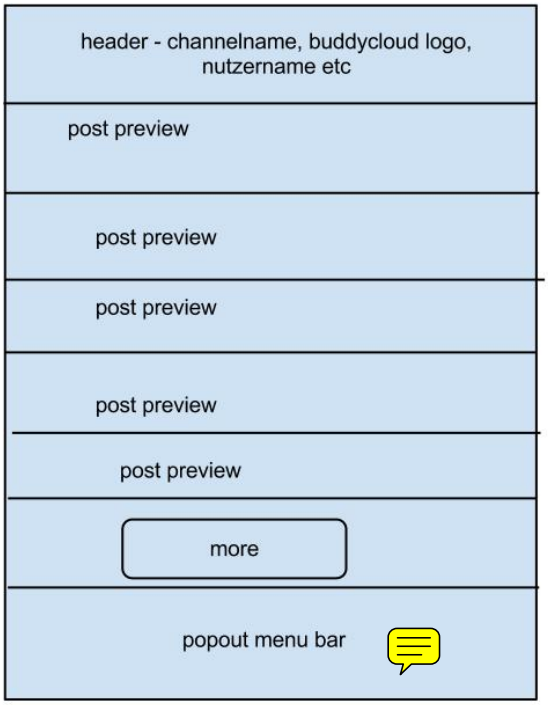


loading screen with buddycloud logo, maybe welcome text, basic buddycloud design, fixed design later



Log in Screen is shown after welcome screen, allowes user to enter jid + passwort to connect to his home server.
Basic buddycloud logo on top, standart buddycloud design in background
Button: Send --> login --> fail or succes
succes: next page: own channel
fail: add fail-reason at bottom of page

Screen is shown on first startup or after log-out, account information is stored encrypted, user needs to privide full jid for his channel and homeserver



HEADER - Picture of own Channel, after log-in users main channel, else current channel, subscribe button
on touch: detailed channel information dropdown

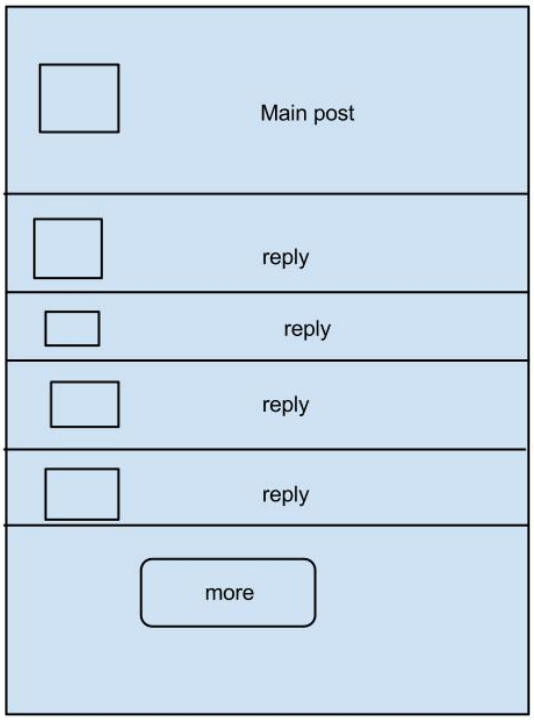
DETAILED CHANNELINFORMATION - Number of followers, likes, maybe important posts, related channels, friends

POSTPREVIEWS - load 10 posts on start, load 10 more on every click on MORE, postpreviews have the picture of the author, preview text, # of answers, likes,follows
on click - switch to view post OR inline view

MORE - load 10 more postspreviews

POPOUT MENU BAR - own channel, (browse back/forward), (more)

PICTURE - redirect to users main channel



MAIN POST - on top, picture of author, meta data (time, # answers, loves etc.), button "love", "reply", post content

REPLY(post) - 10 replys loaded at start, each reply is a ~250 char preview of the post, button "love", "reply", metadata, picture of author, touch/click will expend to full size
-videos/links open in browser?
-pictures embedded

MORE - load 10 more replies

PICTURE - on touch/click load authors channel

REPLY(button) - Opens Texteditor / WYSIWYG, (format post, clean), send button will send the post to the server

(EDIT) - Edit an existing post

MENU BAR - Like Channel

(SCREEN) - drag right/lift -> back to channel/ Extended Option Screen (frind channel, change profile, etc.)

Optionally dragging the screen left/right can open a new screen/activity to browse for other channels, friends, etc.

For posts, the basic prototype is not supposed to support any media files, videos will be opened in a browser if its a link. The First prototypic implementation is not supposed to support any caching, but caching can be added at any time and place during the project. A storage feature that allow the app to store Posts that downloaded before will be an option when there is more time.

Project Scope

The following steps need to be done during the project:

- getting familiar with Android and the Buddycloud API
- getting familiar with best-practice
- create repository
- prepare collaboration (redmine, tasks, wiki)
- decide upon milestones/schedule
- create project hierarchy
- create raw activities (stubs)
- connect to the buddycloud server and store Session
- implement session handling(e.g. reconnect etc.)
- connect activities and pass session information
- make activities load information dynamically

For dev only:

1 -- Develop Buddycloud framework for Android

-- API available (via Session Object)

Impl Detail:

- should stick to flat data types and collections to maintain modularity
- no caching in the framework
- setters and getters
  - set Username + password (sessionobject)
  - get status (sessionobject)
  - get channel content (by jid)
  - ….
- life cycle functions
  - onCreate : create Session Object with domain / domain + account info
  - startConnection
- data models

- status
- representation of channel content
- …...

2 -- Concrete data model

- Account information
- Channel
    - -metadata
    - -channel node
- items (for every type)
- subscription
- media items (for every type)

3 -- Basic Activities

- log-in
- view channel
- view post - inline
- post
- (search - own screen)
- (profile/channel options)

4 -- Data Storage

- caching/permanent storing

all data can be permanently saved in an SQLite database across different user sessions (need to link data to user or delete content after log-out), maybe delete older entries. SQLite is optional and not part of the basic project scope. Alternatively all data can be cached during one user session (in memory)

Data to Cache(items in brackets are still to be discussed)

-(channels)	- memory/persistent?
-(subscriptions)	- memory/persistent?
-(metadata)	- memory/persistent?
- posts	- memory/persistent?
- (mediaitems)	- memory/persistent?

5 -- Dataflow

- Pass session object between activities
- Pass cached objects

Time Schedule:

sum: 10 weeks

19.11-2.12 (2 weeks)

-- get familiar with Android,

-- design of the application (use case of each feature, activities division, data exchange format, communication methods)
-- basic implementation of UI

3.12-9.12(1 week)
-- implementation of the Login function and if possible, a common used authenticating function can be called anywhere in the app

10.12-22.12(2 weeks)
-- implementation and test of the Post getting and displaying feature.

========== 2013 ==========

07.01 milestone - presentation

08.01 - 20.01(2 weeks)
-- implementation and test of the Post Sending feature.

21.01-27.01(1 week)
-- implementation and test of the subscription feature

28.01-10.02(2 weeks)
-- bug fixing and testing

11.02 final presentation